УНИВЕРСИТЕТ ИТМО

**К.И. Салахутдинова**

# МЕТОДЫ ИДЕНТИФИКАЦИИ ОТКРЫТОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ЧАСТЬ I

**Санкт-Петербург**
**2020**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

**К.И. Салахутдинова**

# МЕТОДЫ ИДЕНТИФИКАЦИИ ОТКРЫТОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ЧАСТЬ I

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 10.03.01 Информационная безопасность в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

УНИВЕРСИТЕТ ИТМО

**Санкт-Петербург**
**2020**

Салахутдинова К.И. Методы идентификации открытого программного обеспечения. Часть I – СПб: Университет ИТМО, 2020. – 37 с.

Рецензент(ы):
Лебедев Илья Сергеевич, доктор технических наук, профессор, заведующий лабораторией интеллектуальных систем, ФГБУН Санкт-Петербургский институт информатики и автоматизации Российской академии наук, (СПИИРАН);

Учебно-методическое пособие содержит теоретический и практический материал по существующим методам идентификации программного обеспечения, установленного на электронные носители информации. Рассматривается необходимость проведения аудиторских мероприятий по выявлению несанкционированно установленного программного обеспечения в целях обеспечения информационной безопасности. Наряду с известными алгоритмами рассматриваются новые подходы. Пособие предназначено для студентов бакалавриата, осуществляющим подготовку по направлению 10.03.01 «Информационная безопасность», руководителям и специалистам служб по защите информации.

УНИВЕРСИТЕТ ИТМО

**Университет ИТМО** – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

© Университет ИТМО, 2020
© Салахутдинова К.И., 2020

# Содержание

Введение ..... 4

Обзор проблематики современных подходов к идентификации программного обеспечения ..... 5

Угрозы со стороны несанкционированно установленного открытого программного обеспечения ..... 5

Модель угроз информационной безопасности при обработке информации конфиденциального характера в информационной системе ..... 8

Модель вероятного нарушителя информационной безопасности и оценка его возможностей ..... 11

Обзор существующих методов идентификации программного обеспечения ..... 14

Методы идентификации, основанные на статическом характере анализа характеристик программного обеспечения ..... 16

Методы идентификации, основанные на динамическом характере сбора характеристик программного обеспечения ..... 17

Методы идентификации, основанные на сборе информации путем обращения к встроенным функциям операционной системы или задействования собственного программного агента ..... 18

Формирование образа программы ..... 24

Модель представления программного обеспечения ..... 24

Модель представления исполняемого файла ..... 25

Анализ особенностей ассемблерного кода программного обеспечения . 26

Аудит электронных носителей информации для повышения информационной безопасности автоматизированных систем ..... 29

Список сокращений ..... 322

Список литературы ..... 33

# Введение

Учебно-методическое пособие предназначено для студентов бакалавриата, осуществляющих подготовку по направлению 10.03.01 «Информационная безопасность», и может быть использовано преподавателем в дисциплинах «Инженерно-техническая защита информации», «Комплексное обеспечение информационной безопасности», «Управление информационной безопасностью» и других, где производится рассмотрение организационных и технических мер по обеспечению информационной безопасности в Университете ИТМО.

Материалы, приведенные в пособии, основаны на диссертационном исследовании и позволят студентам получить теоретические знания в рамках решения задачи автоматизированного процесса аудита электронных носителей информации с целью выявления несанкционированно установленного программного обеспечения, нарушающего установленную политику безопасности организации.

В первой главе рассматривается предметная область, задача идентификации программного обеспечения, актуальность и т.п. Приводится рассмотрение моделей угроз и нарушителя. Даются краткие описания возможных уязвимостей, угроз, ущерба.

Во второй главе производится обзор современных подходов к идентификации программного обеспечения, приводится анализ отечественной и зарубежной литературы в данной области. Рассматриваются положительные и отрицательные качества представленных методов.

В третей главе описывается модель представления программного обеспечения и, в частности, модель представления исполняемого файла. Рассматриваются способы обработки программы и дальнейшее формирование ее образа в виде вектора значений выделенного признакового пространства. Также приводится рассмотрение влияния аудита электронных носителей информации для повышения информационной безопасности автоматизированных систем (АС).

Для усвоения учебного материала при самостоятельной работе студентов в пособии изложены систематичные и теоретически обоснованные знания, а в конце каждой главы располагается список контрольных вопросов. Ориентировочные затраты времени на изучение теоретического материала составляются в соответствии с программой изучаемых дисциплин.

В конце приведен список литературы, рекомендуемой автором для более глубокого освоения рассматриваемой темы.

# Обзор проблематики современных подходов к идентификации программного обеспечения

Свободно распространяемое программное обеспечение (ПО) является неотъемлемой частью современных информационных систем, эксплуатируемых в различных секторах экономики, оно позволяет расширить возможности по осуществлению процессов анализа, управления, принятия решений.

Применение открытого ПО обуславливает необходимость разработки дополнительных методов, систем и средств защиты информации. Возможные дефекты программного обеспечения, наличие недекларированных возможностей, нелегальное использование интеллектуальной собственности, применение специальных программ, направленных на преодоление установленной защиты, могут привести к росту числа уязвимостей и повлиять на информационную безопасность систем. В связи с этим возникает необходимость решения ряда задач идентификации, верификации и валидации программного обеспечения [1].

### *Угрозы со стороны несанкционированно установленного открытого программного обеспечения*

Будем понимать под идентификацией программного обеспечения процедуру построения некоторой информативной модели программы (*сигнатуры*) по выбранному признаковому пространству, характеристики которой позволяли бы с заданной точностью найти соответствие между рассматриваемой (*идентифицируемой*) и конкретной программой. Другими словами, идентифицировать исполняемый файл будет означать распознать его как той или иной программы.

Под открытым программным обеспечением понимаются исполнимые и компонуемые файлы 32х и 64х разрядностей формата ELF [2] (рисунок 1) в Linux операционных системах (ОС) для архитектур процессоров x86 и x86-64.

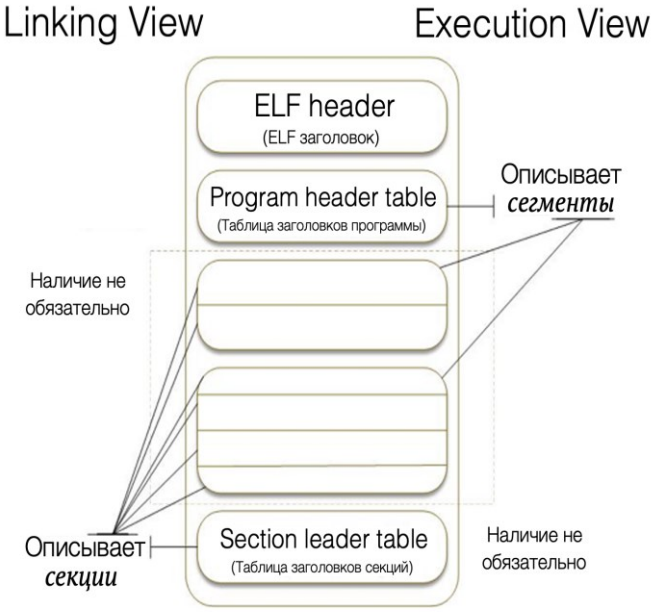


*Рисунок 1– Представление ELF-файла*

Стремительный прогресс информационных технологий, их повсеместное внедрение во все сферы человеческой деятельности приводит к тому, что современная организация бизнес процессов становится полностью зависима от надлежащего функционирования информационных систем (ИС). Область, отвечающая за безопасность информации обрабатываемой в таких системах и ее ресурсов, является областью информационной безопасности (ИБ).

Понятие ИБ трактуется различными ГОСТами по-разному, может определяться как процесс: «защита конфиденциальности, целостности и доступности информации; кроме того, сюда могут быть отнесены и другие свойства, например, аутентичность, подотчетность, неотказуемость и надежность» [3], как свойство: «свойство информации сохранять конфиденциальность, целостность и доступность» [4], как состояние: «состояние защищенности информации [данных], при котором обеспечены ее [их] конфиденциальность, доступность и целостность» [5]. Однако из всех определений вытекает главная цель информационной безопасности – обеспечение трех наиболее важных свойств защищаемой информации: *конфиденциальности, целостности и доступности.*

Основным документом, описывающим порядок и принципы обеспечения информационной безопасности в организации, является политика ИБ, определение которой вытекает из двух понятий ГОСТа Р ИСО/МЭК 27002-2012, а именно, информационной безопасности, «Защита конфиденциальности, целостности и доступности информации; кроме того, сюда могут быть отнесены и другие свойства, например, аутентичность, подотчетность, неотказуемость и надежность» [3] и политики, «Общее намерение и направление, официально выраженное руководством». Таким образом, *политика информационной безопасности* является совокупностью документированных управленческих решений, направленных на защиту необходимых свойств информации и ассоциированных с ней ресурсов.

Дополнительно необходимо отметить, что представляет собой понятие конфиденциальной информации, ведь по законам Российской Федерации, защите [6] подлежит секретная [7] и конфиденциальная информация [8]. Ранее понятие конфиденциальной информации давалось в ныне утратившем силу ФЗ от 20 февраля 1995 г. № 24-ФЗ: «документированная информация, доступ к которой ограничивается в соответствии с законодательством Российской Федерации» [9], теперь же в соответствии с ФЗ от 27 июля 2006 г. № 149-ФЗ понятие конфиденциальной информации вытекает из двух определений: конфиденциальность информации, «обязательное для выполнения лицом, получившим доступ к определенной информации, требование не передавать такую информацию третьим лицам без согласия ее обладателя», и информация, «сведения (сообщения, данные) независимо от формы их представления» [6]. Таким образом, *конфиденциальная информация* – это сведения, независимо от

формы их представления, которые не могут быть переданы лицом, получившим доступ к данным сведениям, третьим лицам без согласия их правообладателя.

При взаимодействии пользователя с информационной системой организации, ее ресурсами и обрабатываемой внутри информацией наступает потребность в регулировании такого взаимодействия, реализуемого за счет политики ИБ, мер и средств по обеспечению безопасности [10]. В настоящем пособии идентификация ПО рассматривается как техническая мера обеспечения безопасности, подкрепляющая собой организационную меру, часто выраженную в политике безопасности организации о запрете на несанкционированное установление ПО.

С каждым годом все более доступным становится доступ к различным страницам, сайтам сети Интернет, ориентированным на информирование интернет-пользователей о существующих уязвимостях операционных систем, программного обеспечения и эксплуатации нетривиальных способов обхода установленных на рабочих местах систем защиты информации. Также не малую роль играет то, что Linux ОС и большое количество программного обеспечения для них представляют собой свободное программное обеспечение, пользователи которого, в частности, имеют права на его свободное использование и, что самое важное, модификацию. Все вышеописанное ведет к тому, что аудит электронных носителей информации на предмет выявления несанкционированно установленного ПО становится важной и все более актуальной задачей. Аудит электронных носителей информации позволяет выявить нарушения установленной политики безопасности при обработке конфиденциальной информации.

Действия пользователей АС, направленные против установленной политики безопасности в организации, способны стать источником роста числа уязвимостей системы и повлиять на ее ИБ. Причиной этого являются возможные дефекты ПО, наличие недекларированных возможностей, нелегальное использование интеллектуальной собственности, а также использование специальных программ, направленных на преодоление установленной защиты либо противоправных действий внутри сети Интранет или Интернет. Последнее особенно актуально в области преступлений, связанных с компьютерной информацией [11].

Во второй главе будет приведен обзор существующих подходов к идентификации установленного программного обеспечения, представлены их ограничения, достоинства и недостатки.

***Модель угроз информационной безопасности при обработке информации конфиденциального характера в информационной системе***

В соответствии с описанными выше определениями ИБ, в обобщенном виде формулирующими поддержание наиболее важных свойств защищаемой информации: конфиденциальности, целостности и доступности, приведем описание объекта и целей реализации угроз относительно задачи данного исследования (рисунок 2).

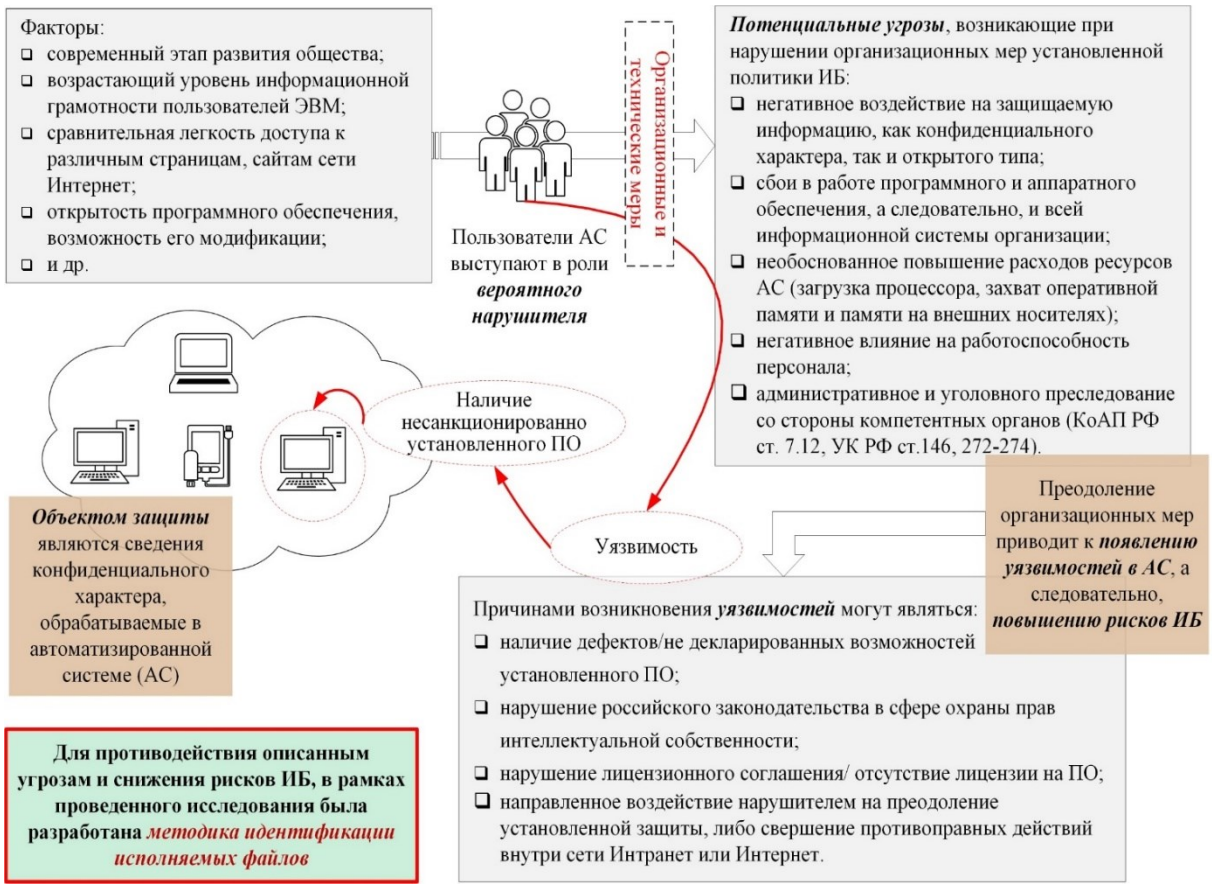


*Рисунок 2 – Взаимосвязь факторов, угроз и появление новых уязвимостей при несанкционированной установке ПО*

По определению, *объектом защиты информации* является «информация или носитель информации, или информационный процесс, которые необходимо защищать в соответствии с целью защиты информации» [5], таким образом, со стороны обеспечения информационной безопасности ***объектом защиты*** являются сведения конфиденциального характера, обрабатываемые в автоматизированной системе, для которых необходимо предотвратить утечку, несанкционированные и непреднамеренные воздействия. В качестве объекта может выступать сама автоматизированная система, ее ресурсы, электронные носители информации, и программное обеспечение, представляющее собой потенциальный источник уязвимостей системы [12].

***Целью реализации угроз*** является осуществление личных интересов пользователем АС посредством преднамеренной установки несанкционированного ПО, способного привести к материальному или моральному ущербу организации.

В качестве ***потенциальных угроз***, способных привести к нижеописанному ущербу, выступают следующие угрозы, возникающие при нарушении организационных мер установленной политики ИБ организации:

– негативное воздействие на защищаемую информацию, как конфиденциального характера, так и открытого типа;

– сбои в работе программного и аппаратного обеспечения, а следовательно, и всей ИС организации;

– необоснованное повышение расходов ресурсов АС (загрузка процессора, захват оперативной памяти и памяти на внешних носителях);

– негативное влияние на работоспособность персонала;

– административное и уголовное преследование со стороны компетентных органов.

Преодоление организационных мер приводит к ***появлению уязвимостей в АС***, а следовательно, ***повышению рисков ИБ*** организации. Несанкционированно установленное ПО может не только стать лазейкой для противоправных действий нарушителя ИБ, но также привести к критическим ошибкам системы, способным остановить отлаженные бизнес-процессы, замедлить работу АС, повлиять на работоспособность персонала путем переноса внимания работника от непосредственных обязанностей, деморализации коллективных отношений и т.д. Так, использование нелицензионного ПО нарушает законодательство Российской Федерации и влечет наложение административной (если стоимость контрафактных экземпляров программ ЭВМ составит менее 50000 рублей) или уголовной ответственности (если стоимость контрафактных экземпляров программ ЭВМ составит более 50000 рублей) за нарушение авторских и смежных прав, которая предусматривается статьей 146 Уголовного кодекса Российской Федерации (УК РФ). При этом ответственность за противоправное использование ПО, в основном, возлагается на руководителя организации, принявшего управленческое решение по его использованию или же физическое лицо, самовольно установившее такое ПО. Отдельно необходимо отметить, что преступления по статье 146 УК РФ относятся к преступлениям публичного обвинения и не требуют подачи заявления с потерпевшей стороны. Ответственность же варьируется от административного штрафа в размере 1500-2000 рублей для физического лица до лишения свободы на срок до 6 лет со штрафом в размере до 500000 рублей. Так, согласно исследованию BSA | The Software Alliance [13], было отмечено, что использование нелицензионного ПО в 2017 году составило 62%. При этом коммерческая стоимость нелицензионного ПО составила

1,291 миллиарда долларов США. По данным отчета о видах наказания по наиболее тяжкому преступлению (без учета сложения) за 2017 год к уголовной ответственности по статье 146 УК РФ было привлечено 482 человека [14].

Причинами возникновения ***уязвимостей*** могут являться:

– наличие дефектов / недекларированных возможностей установленного ПО;

– нарушение российского законодательства в сфере охраны прав интеллектуальной собственности;

– нарушение лицензионного соглашения / отсутствие лицензии на ПО;

– направленное воздействие нарушителем на преодоление установленной защиты, либо свершение противоправных действий внутри сети Интранет или Интернет.

В качестве ***потенциального ущерба*** от реализации угроз может быть причинен:

– моральный и материальный ущерб деловой репутации организации, возникающий в результате выявления надзорными органами нарушений использования чужой интеллектуальной собственности;

– моральный, физический или материальный ущерб, связанный с разглашением персональных данных отдельных лиц, который может возникнуть в результате использования ПО, имеющего дефекты или направленного на преодоление установленной защиты;

– материальный ущерб от разглашения защищаемой информации, который может возникнуть в результате возникновения уязвимостей в АС при использовании ПО, имеющего дефекты или направленного на преодоление установленной защиты;

– материальный ущерб от необходимости восстановления нарушенных защищаемых информационных ресурсов, доступ или целостность которых может быть нарушена;

– моральный и материальный ущерб от дезорганизации деятельности организации, вызванный уменьшением работоспособности пользователей АС;

– материальный и моральный ущерб от нарушения международных отношений, который может возникнуть в связи с нанесенным ущербом деловой репутации организации.

Источниками угроз являются пользователи АС, реализующие деятельность по преднамеренной несанкционированной установке ПО с целью извлечения своей выгоды. Умысел может носить разнообразный характер – как подстраивайте АС «под себя», так и намеренная

дестабилизация организованной системы защиты, и добывание защищаемой информации.

### *Модель вероятного нарушителя информационной безопасности и оценка его возможностей*

По признаку принадлежности к АС организации все нарушители будут являются внутренними, в силу того, что у них имеются некоторые права допуска в помещения и доступа к ЭВМ.

Естественно, по отношению к АС организации нарушителем может являться и внешнее лицо, но, если предполагать, что ИБ организации находится в устойчивом состоянии до того, как произойдет инцидент с несанкционированной установкой ПО, тогда можно считать, что для преодоления установленных средств защиты информации (ЗИ) внешнему нарушителю необходимы вначале действия внутреннего, которые повлекут за собой появление ранее не рассматривавшихся уязвимостей.

Возможности внутреннего нарушителя ограничены комплексом действующих в пределах контролируемой зоны режимных и организационно-технических мер, направленных, на пресечение действий пользователей АС по несанкционированной установке ПО.

Однако в зависимости от многих факторов (например, количество денежных средств, выделяемых на организацию ИБ; наличие и уровень подготовки специалистов по ИБ; размер штата и ресурсов в организации; добросовестность и уровень подготовки персонала в сфере ИБ; и т.п.), организационно-технические меры могут быть недостаточными, и в таком случае у внутреннего нарушителя появляется возможность преодоления установленной политики безопасности.

Исходя из особенностей функционирования различных типов организаций, обрабатывающих информацию, подлежащую защите, к ***внутренним нарушителям*** можно отнести следующие общие категории персонала [15]:

– лица, имеющие непосредственный доступ к АС, но не имеющие доступа к защищаемой информации, при этом действия пользователей, относящихся к данной категории, способны привести к дестабилизации комплекса защиты информации и увеличению рисков ИБ в сегменте локальной сети, где они расположены (категория 1);

– лица, имеющие непосредственный доступ к АС и имеющие доступ к защищаемой информации, а также удаленный доступ по локальным и распределенным информационным системам, при этом действия пользователей, относящихся к данной категории, в дополнение к вышеописанной первой категории, способны привести к нежелательному воздействию на свойства защищаемой информации (категория 2);

– лица, имеющие непосредственный доступ к АС и имеющие полномочия администратора безопасности сегмента информационной системы организации, здесь действия пользователя имеют более весомый характер воздействия на состояние защищенности информации, в дополнение руководящие позиции попадают под ответственность о защите интеллектуальной собственности (категория 3);

– лица, имеющие непосредственный доступ к АС и имеющие полномочия системного администратора или администратора безопасности информационной системы организации, здесь следует учитывать, что данная категория пользователей обладает бóльшими правами и возможностями, знаниями относительно технических средств обработки информации, а также ресурсами программного обеспечения (категория 4);

– лица, занимающиеся разработкой, сопровождением и ремонтом технических средств в информационной системе организации, могут являться штатными/внештатными сотрудниками и производить несанкционированные действия по установке ПО (категория 5).

Стоит также отметить, что к защищаемой информации относятся не только сведения, попадающие под указ Президента РФ №188 [8], но и общедоступная информация, ограничения доступа к которой быть не должно [6], а также информация, дестабилизирующее воздействие на которую (уничтожение, модификация, искажение, копирование, блокировка) может привести к моральному или материальному ущербу организации и ее деловой репутации.

Предполагается, что пользователи всех категорий способны реализовывать угрозы ИБ, используя стандартное оборудование, расположенное в локальной сети организации, воздействие через которое способно повлиять на всестороннюю комплексность ИБ.

Кроме того, предполагается, что лица категории 3 и 4 хорошо знакомы с протоколами и алгоритмами, задействованными в информационной системе организации, и могут располагать специализированным оборудованием и программным обеспечением, позволяющим беспрепятственно производить несанкционированную установку ПО.

Для категорий 1-4 справедлива угроза снижения работоспособности персонала.

Предполагается, что для крупных организаций лица категории 3 и 4 проходят тщательную проверку со стороны руководства организации и являются доверенными. К сожалению, для более малых организаций такой тщательной проверки на квалифицированность и добропорядочность не производится, поэтому данные категории наравне с остальными могут относиться вероятным нарушителям.

Возможны следующие способы реализации угроз ИБ:

– несанкционированный доступ к защищаемой информации за счет использования функциональных особенностей несанкционированно установленного ПО, либо наличии в нем недекларированных возможностей либо уязвимостей старых/новых версий;

– несанкционированный доступ к средствам обеспечения защиты информации;

– негативное воздействие на программно-технические компоненты ИС;

– использование программ игрового, информационного, развлекательного характера или предназначенных для обмена разного вида сообщениями;

– отправка сведений о нелицензионном использовании ПО компетентным органам или отрицательное заключение по результатам их аудита.

Таким образом, для противодействия описанным угрозам и снижения рисков ИБ с учетом возможностей нарушителя и возможных убытков необходимо производить более тщательный и своевременный аудит электронных носителей информации на наличие несанкционированно установленного ПО.

*Контрольные вопросы:*

1. Каким образом программное обеспечение может нанести вред информационной системе организации?
2. Каково определение информационной безопасности в соответствии с ГОСТами?
3. Что является целью идентификации невредоносного программного обеспечения?
4. Какие причины возникновения уязвимостей, связанные с невредоносным программным обеспечением, вы знаете?
5. Какая из категорий нарушителей является потенциально наиболее опасной?

# Обзор существующих методов идентификации программного обеспечения

В задаче идентификации объект $o \in O$ (где $O$ – множество объектов) описывается при помощи некоторого выделенного признакового пространства $f_1, f_2, \ldots, f_n$ (где $i = 1, 2, \ldots, n$ – число градаций признака и значение для $f_n \in D_f$ – множеству допустимых значений признака), способного охарактеризовать основное свойство объекта $p(o)$, т.е. его класс, и записывается в виде вектора значений признака

$$o = (f_1(o), f_2(o), \ldots, f_n(o)),$$

именуемого в дальнейшем сигнатурой программы. Предполагается, что существует функциональная связь между признаками $f_1, f_2, \ldots, f_n$ и основным свойством объекта $p(o)$, позволяющая определить значение свойства $p(o)$. В данном исследовании в качестве основного свойства выступает имя программы, полученное по ранее составленной информации о других объектах $o_1, o_2, \ldots, o_n$ (называемых обучающей выборкой) и их свойствах $p(o_1), p(o_2), \ldots, p(o_n)$.

Идентификация происходит путем применения выбранного алгоритма сравнения двух векторов значений признака объектов: $o$ – идентифицируемого и $o'$ – эталонного.

Различают следующие типы признаков [16]:

– количественные признаки – признаки, измеренные в определенных метрических (числовых) шкалах;

– качественные – признаки, измеренные в неметрических шкалах и используемые для выражения терминов и понятий, не имеющих цифровых значений.

Также, в зависимости от множества допустимых значений признака, выделяют следующие типы [17]:

– бинарный – принимающий значение нуля или единицы, $D_f = \{0,1\}$;

– номинальный – где множество допустимых значений признака конечно, $|D_f| = const$;

– порядковый – где множество допустимых значений признака конечно и упорядочено, $|D_f| = const$ и $D_f = \{d_f \mid d_{fj} \leq d_{fj+1}, j = 1, \ldots, const\text{-}1\}$;

– количественный – где в качестве допустимых значений признака выступает множество действительных чисел $D_f \in \mathbb{R}$.

В свою очередь, представление признаков может носить разнообразный характер, в частности, являться перечислением признаков, векторным описанием файла, или же иметь графическое представление, записанное в древовидной или графовой модели. В данном пособии будут в основном рассмотрены номинальные (номинативные) признаки – это

признаки, измеренные в неметрической шкале наименований, представляющие собой набор числовых характеристик.

На сегодняшний день существует ряд подходов к идентификации программного обеспечения на электронных носителях информации. Процесс распознавания (или идентификации) программы можно разделить на два этапа: первый - выделение признакового пространства и второй - применение метода распознавания программы.

Подходы к распознаванию программ могут быть основаны как на структурной характеристике файла, описывающей его содержимое, так и на семантической составляющей, характеризующей смысл анализируемого кода программы. К последним относится интеллектуальная обработка данных, в которую входят [18]:

– формальные методы – оперирующие фиксированной моделью объекта и использующие предопределенные алгоритмы обработки;

– эвристические методы – способные, в некоторой степени, воспроизвести процесс познания человека; они направлены на выявление неизвестных объектов.

Методы интеллектуальной обработки данных, как правило, включают проверку «корректности» исполняемых файлов, вычисление статистического распределения инструкций процессора или энтропии кода, а также выявление последовательностей инструкций, характерных для вредоносных программ. В связи с этим использование данной группы методов в целях идентификации невредоносного программного обеспечения не представляется возможным (достаточно эффективным).

Ниже представлены наиболее распространенные подходы к задаче идентификации исполняемого файла.

*Таблица 1 – Подходы, используемые для идентификации программ*

| Способы сбора характеристик программы | Методы распознавания программ |
|---|---|
| – По *статическим* характеристиками рассматриваемого объекта<br>– По *динамическим* характеристиками рассматриваемого объекта<br>– Задействующие *встроенные функции операционной системы* или функциональность внедряемого программного агента | – Задание метрики схожести<br>– Статистический анализ<br>– Машинное обучение<br>– Эвристический анализ |

***Методы идентификации, основанные на статическом характере анализа характеристик программного обеспечения***

*Статические методы* [19,20] оперируют двоичным образом программы, хранящемся на электронном носителе информации, будь то внешний жесткий диск или оперативная память компьютера. Данный метод подразумевает под собой работу с исполняемым файлом или его дампом памяти как с массивом побайтовой последовательности.

Статические методы, по способу сбора характеристик программы, являются наиболее распространенными. Данный класс методов подразделяется на *несигнатурные* (методы проверки целостности), такие как сравнение с полной копией данных, сравнение контрольных сумм, контроль CRC (Cyclic Redundancy Check), хеширование, имитовставка, цифровая подпись, и *сигнатурные* (методы сравнения признаковых последовательностей), такие как выделение «магического числа», сравнение характерных последовательностей байтов и др.

В средствах защиты информации активно используется сигнатурный (синтаксический) метод, заключающийся в сравнении байтовых последовательностей с эталонной сигнатурой.

Идентифицировать файл можно с разной степенью конкретности, так, самой простой идентификацией будет являться определение расширения файла по его сигнатуре, известной также как «магическое число», представляемое в виде строго определенного набора байт, как правило, внесенных в начало тела файла. Так, например, для ELF файла эта последовательность задана следующим набором байт – 7F 45 4C 46; а для EXE файлов – 4D 5A.

Несигнатурные методы также популярны в настоящее время, но сфера их применения ограничена, и ниже будут описаны их достоинства и недостатки применительно к задаче идентификации ПО.

Одним из первых и наиболее простых методов обеспечения целостности файла является разработка метода контрольной суммы. Первоначально под контрольной суммой понималось некоторое число, занимающее позицию младшего разряда байта и вычисляемое при помощи сложения остальных 7 бит. Сегодня под данное определение попадает огромное число разнообразных алгоритмов, некоторые из которых имеют широкое применение, а некоторые используются только для специфических задач. Самыми известными и популярными алгоритмами являются SHA-1, SHA-2, SHA-3, CRC и MD5, при этом стоит отметить, что последние два алгоритма применяются все реже, т.к. не способны обеспечить достаточный уровень стойкости к коллизиям.

Хеширование является простейшим подходом к сравнению файлов [21,22]. При вычислении хеш-функции в качестве исходных данных принимается исходная двоичная последовательность файла, далее в зависимости от выбранного алгоритма она разбивается на блоки

фиксированной длины с последующим вычислением значения хеш-функции.

Достоинством хеш-функции является ее простота, скорость выполнения операции и размер получаемого значения хеш-функции, который всегда имеет одну и ту же длину. Однако, существенным недостатком в задаче сравнения двух файлов является то, что вычисляемое значение хеш-функции зависит от модификации, произведенной над файлом, при этом оно ни в коей мере не зависит от степени произведенного вмешательства над оригинальным файлом. Т.е. если сравнивать значение хеш-функции оригинального файла со значением хеш-функции оригинального файла с измененным хотя бы одним битом – эти значения будут совершенно различными.

Чтобы снизить влияние модификаций и сравнивать схожие файлы, был предложен подход с использованием контекстно-зависимого кусочного хеширования (Context Triggered Piecewise Hashing), который использует плавающее окно с фиксированным значением шага, таким образом, контекстно-зависимое кусочное хеширование представляет собой конкатенацию значений хеш-функций блоков файла. Данный подход использовался в работах [23-25]. Модификации алгоритма выбора длины окна, шага и границ образуемых блоков приводят к существенному улучшению при идентификации схожих файлов, однако остается ограничение на количество возможных вносимых изменений в оригинальный файл.

Цифровая подпись в исполняемых файлах используется для верификации программы ее автором, позволяя идентифицировать издателя файла (подлинность файла) и определить, не подвергался ли он изменениям (целостность файла), тем самым гарантируя безопасность программы для АС. Таким образом, если файл содержит некорректную цифровую подпись (или ее нет совсем), то это может означать, что данный файл опубликован ненадежным издателем или был изменен [26], однако существуют способы обхода проверки цифровой подписи и запуска недостоверного программного обеспечения [27].

### *Методы идентификации, основанные на динамическом характере сбора характеристик программного обеспечения*

*Динамический анализ* основан на исполнении анализируемой программы. Отсутствие каких-либо предположений о ходе исполнения программы, а также проверка её в процессе выполнения (или сразу по окончании) является значимым преимуществом динамического анализа. Очевидным требованием, предъявляемым при реализации динамического анализа, является проведение динамического анализа с наименьшим влиянием на ход исполнения программы. При определённых условиях на

детерминированность программы динамический анализ позволяет избежать проблемы ложных срабатываний.

Большинство современных вредоносных программ используют методы запутывания (обфускации), чтобы помешать статическому анализу двоичных файлов, в частности бинарные упаковщики. Поэтому динамический анализ таких вредоносных программ зачастую бывает намного эффективнее статического анализа. Мониторинг поведения программ в ходе их исполнения включает, например, сбор выполняемых программой бинарных операций и предлагает потенциально более глубокое понимание самого кода. Сегодня подходы такого типа имеют свои ограничения, например – может быть трудно вызвать определенное поведение программ, а некоторые из них могут требовать определенные условия окружающей среды, тем не менее, использование динамического анализа считается более эффективным по сравнению с использованием только лишь статических подходов. По этой причине, динамическому анализу программ уделяется большее внимание в научном сообществе. Системы анализа, такие как CWSandbox [28], Anubis [29], BitBlaze [30], AVG [31] и Advanced Threat Protection [32], выполняют запуск образцов вредоносных программ в защищенной среде [33] и отслеживают их поведение для анализа и разработки защитных механизмов.

Основными считаются подходы по [34]:

– выявлению последовательности необычных путей исполнения кода программы или вызываемым системным функциям (API, system calls, Windows native API) [35];
– выявлению нанесенных повреждений окружающей среде в изолированной системе [36];
– отслеживанию попыток получения высоких привилегий [37];
– выявлению изменений параметров вызываемых функций [38];
– определению способов обработки данных программой [39];
– анализу трассировки машинных инструкций программы [40].

Динамический анализ является более эффективным в плане анализа программы, однако для задачи идентификации большого числа версий невредоносных программ его применение становится нецелесообразным, поскольку его главный минус связан с большими временными затратами, требуемыми на выполнение и тщательное исследование программ.

***Методы идентификации, основанные на сборе информации путем обращения к встроенным функциям операционной системы или задействования собственного программного агента***

В настоящее время множество компаний обращаются к использованию систем управления ИТ-активами (ITAM), которые представляют собой комплексные решения, нацеленные на физический

учёт, финансовый контроль и соблюдение контрактных обязательств, связанных с ИТ-активами, на протяжении всего их жизненного цикла [41,42]. Здесь под ИТ-активами подразумеваются все аппаратные и программные элементы ИТ-инфраструктуры, обеспечивающие деятельность бизнес-среды.

В свою очередь ITAM подразделяется на управление аппаратными активами (HAM), охватывающее управление материальными составляющими ИТ-инфраструктуры: пользовательские компьютеры, сервера, телефоны и т.д.; и на управление программными активами (SAM), охватывающее управление нематериальными составляющими ИТ-инфраструктуры: программное обеспечение, лицензии, версии, конечные точки инсталляции и т.д.

На рынке услуг представлено немало решений [43], позволяющих идентифицировать программные активы, управлять учетом, а также производить контроль их изменений и др. Из числа наиболее известных программных продуктов можно выделить:

1) Работающие на основе агента:
   - Samanage – поддерживает работу с Windows, Linux, Mac OC [44].
   - Kaspersky Systems Management – поддерживает работу с Windows, Linux, Android, iOS [45].
2) Работающие без агента:
   - Microsoft Assessment and Planning Toolkit – поддерживает работу с Windows OC [46].
3) Работающие как с агентом, так и без него:
   - AIDA64 – поддерживает работу с Windows, Linux, Android [47].
   - Lansweeper – поддерживает работу с Windows, Linux, Mac OC и устройствами, поддерживающими IP адресацию [48].

Агентом является программа, устанавливаемая на компьютер пользователя АС и ведущая скрытую деятельность по выполнению своих функций, в частности инвентаризации ПО. Недостатком данного подхода является необходимость установки агента на каждый исследуемый компьютер, однако преимуществом агента является возможность его установки на корпоративные ноутбуки и смартфоны, используемые пользователями за пределами внутренней сети организации. Таким образом, решения, работающие на основе агента, имеют самый высокий уровень детализации, могут быть использованы для вмешательства в компьютер пользователя. Использование подхода без установки агента не требует наличия подходящих для данного агента систем, его обновлений, не повышает производительные затраты ресурсов компьютера, а также имеет удобное централизованное управление. Однако уровень детализации получаемой информации ниже, чем при задействовании агента, а также появляется необходимость контроля того, что исследуемые активы доступны для их обнаружения.

Подход к оценке ресурсов с(без) агентом имеет свои преимущества и недостатки, особенно стоит учитывать число самих ресурсов в организации.

Однако отметим, что большая часть предоставляемой информации собирается посредством встроенных технологий инвентаризации программного и аппаратного окружений, операционной системы:

– Windows Management Instrumentation – является одним из базовых инструментариев централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows;

– Active Directory Domain Services – служба каталогов Windows, которая централизованно сохраняет все данные и настройки среды в базе данных;

– SMS Provider – поставщик инструментария управления Windows, назначающий доступ на чтение и запись к базе данных Configuration Manager на сайте сервера;

– lshw – утилита Linux, которая выводит полный список аппаратных компонентов системы вместе с информацией об устройствах;

– dpkg -l – утилита Linux, которая выводит полный список программных компонентов системы;

– и других технологий.

Недостатки такого подхода очевидны. Отсутствие должного уровня оценивания роли информационной безопасности в бизнес-процессах организации приводит к недооценке кадровой политики со стороны руководства при подборе ИТ персонала. В то же время пользователи АС имеют возможность достичь достаточного уровня компетенции в сфере компьютерных технологий, позволяющего им обходить установленные политики безопасности. Все это предоставляет возможность внесения изменений в конфигурационные данные устанавливаемого программного обеспечения.

Ниже на рисунках 3-5 приведены примеры по изменению версий программ (*ABBYY Lingvo* и *Htop*) в операционных системах Windows (версия: 10) и Linux (версия: Ubuntu) соответственно. Из них становятся очевидны пути обмана средств аудита программного обеспечения как для установленного ранее программного обеспечения, так и для устанавливаемого впервые.

*Рисунок 3 – Изменение версии программы ABBYY Lingvo x5 15.0.511 на ABBYY Lingvo x5 15.0.510*

localhost:81/Software/report.aspx

Lansweeper | Search... | Trial license (100 assets). Purchase Lansweeper here.

Dashboard | Assets | Reports | Software | Scanning | Helpdesk | Knowledgebase | Calendar | Deployment | Configuration

Report options
- Export to Excel
- Export to CSV
- Export to XML

Windows
- Windows Software
- Edit Purchased Licenses

Windows Software (94)

| Software | Version | Publisher | Total |
|---|---|---|---|
| 7-Zip 9.20 (x64 edition) | 9.20.00.0 | Igor Pavlov | 1 |
| ABBYY Lingvo x7 | 15.0.512 | ABBYY | 1 |
| Adobe Acrobat Reader DC - Russian | 19.008.20081 | Adobe Systems Incorporated | 1 |
| Adobe Flash Player 15 Plugin | 15.0.0.223 | Adobe Systems Incorporated | 1 |

*Рисунок 4 – Изменение версии программы ABBYY Lingvo x5 15.0.511 на ABBYY Lingvo x7 15.0.512*

Из рисунков 3 и 4 видно, что путем манипуляций в реестре операционной системы Windows версия программы *ABBYY Lingvo* была изменена дважды. Целью данного вмешательства являлось намеренное влияние на результаты выдачи запроса по инвентаризации программного обеспечения с помощью средств Windows Management Instrumentation (WMI) (результат выдачи представлен на рисунке 3) и распространенного программного средства Lansweeper (результат выдачи представлен на рисунке 4).

root@ubunt

File Edit View Search Terminal Help

```
dpkg-query: package 'htop' is not installed
Use dpkg --info (= dpkg-deb --info) to exami
and dpkg --contents (= dpkg-deb --contents)
root@ubuntu:~# dpkg -s htop
Package: htop
Status: install ok installed
Priority: optional
Section: utils
Installed-Size: 216
Maintainer: Ubuntu Developers <ubuntu-devel-
Architecture: amd64
Version: 8.1.0-3
Depends: libc6 (>= 2.15), libncursesw5 (>= 6
```

archive.ubuntu.com/ubuntu/pool/m

**Index of /ubuntu/pool/main/h**

| Name | Last modified | Size |
|---|---|---|
| Parent Directory | | - |
| htop_2.1.0-3.debian.tar.xz | 2018-02-17 23:23 | 8.6K |
| htop_2.1.0-3.dsc | 2018-02-17 23:23 | 1.9K |
| htop_2.1.0-3_amd64.deb | 2018-02-17 23:23 | 78K |
| htop_2.1.0-3_i386.deb | 2018-02-17 23:24 | 83K |
| htop_2.1.0.orig.tar.gz | 2018-02-05 17:08 | 296K |
| htop_2.2.0-1.debian.tar.xz | 2018-05-04 21:05 | 11K |
| htop_2.2.0-1.dsc | 2018-05-04 21:05 | 2.2K |
| htop_2.2.0-1_amd64.deb | 2018-05-04 21:10 | 78K |
| htop_2.2.0-1_i386.deb | 2018-05-04 21:14 | 83K |
| htop_2.2.0.orig.tar.gz | 2018-05-04 21:05 | 285K |
| htop_2.2.0.orig.tar.gz.asc | 2018-05-04 21:05 | 819 |

*Apache/2.4.18 (Ubuntu) Server at archive.ubuntu.com Port 80*

**Details**

| | |
|---|---|
| Version | 8.1.0-3 |
| Updated | Never |
| License | Proprietary |
| Source | htop_2.1.0-3_amd64.deb |
| Download Size | 0 bytes |

*Рисунок 5 – Установка измененной версии программы htop 8.1.0-3 вместо htop 2.1.0-3*

На рисунке 5, в свою очередь, представлена установка программы *Htop* в операционной системе Ubuntu с заведомо измененной версией, исправленной путем вмешательства в конфигурационный файл устанавливаемого пакета *.deb.

В данной главе был произведен обзор существующих методов идентификации программного обеспечения, из которого можно выделить ряд ограничений, не позволяющий использовать их в задаче идентификации версий программного обеспечения:

1) Для статического сбора характеристик файла:

– Методы, основанные на оценке целостности файла, не позволяют исследовать схожесть программ, не задействованных в обучающей выборке.

– Использование особенностей PE формата программ ОС Windows, отсутствующие в ELF-файлах.

– Большинство существующих подходов к классификации исполняемых файлов направлены на бинарное разделение тестовой выборки и выделение класса вредоносных программ.

– Часть подходов рассматривает мульти-классификацию с ограниченным набором классов.

– Работы же, направленные на мульти-классификацию с неограниченным набором классов, обладают низкой точностью идентификации исполняемых файлов.

2) Для динамического сбора характеристик файла:

– Использование данного типа подходов является нецелесообразным по причине задействования огромного числа ресурсов.

– Для выполнения программы бывает необходимо создавать особую среду исполнения.

3) Для сбора информации путем обращения к встроенным функциям операционной системы или задействования собственного программного агента:

– Главным недостатком является сам принцип получения информации с АС, где факт наличия способов изменения информации о программе был проиллюстрирован на рисунках 3-5.

Приведенный анализ методов идентификации различных версий исполняемых файлов позволяет сделать вывод, что идентификация на основе статического анализа характеристик программы и использование статистических и машинных алгоритмов классификации является довольно перспективным направлением деятельности в рамках задачи идентификации невредоносного ПО.

Можно сказать, что в современных условиях не существует решения, позволяющего с достаточной точностью производить идентификацию различных версий ПО в АС в задаче по выявлению факта нарушения

пользователем установленных правил по несанкционированной установке ПО. Отсюда вытекала необходимость в проведении автором данного пособия исследования [1], направленного на достижение максимальной точности идентификации исполняемых файлов, путем совершенствования существующих подходов, поиска новых, ранее не применявшихся решений в области выделения признакового пространства файлов, методов идентификации, а также способов постобработки результатов.

*Контрольные вопросы:*

1. Чем характеризуются различные типы признаков?
2. Каковы отличия подходов, используемых для идентификации программ?
3. Что подразумевает собой метод идентификации, основанный на сборе информации путем обращения к встроенным функциям операционной системы?
4. Какие основные достоинства и недостатки приведенных методов вы можете назвать?
5. Вспомните случай из своей жизни, когда вам приходилось использовать какой-либо из описанных методов.

# Формирование образа программы

### *Модель представления программного обеспечения*

Ознакомившись с основными существующими методами идентификации ПО, сформируем модели представления ПО в целом и для отдельного исполняемого файла. Формализованное представление в виде моделей поможет структурировать и представить анализируемый объект, а также выразить главный принцип авторской методики идентификации ПО не основанном на целостности кода программы.

Идентифицируемый исполняемый файл обладает характеристиками, свойственными некоторой программе. Таким образом, можно детерминировать некоторую программу $P_{\text{этал.},i}$ как совокупность ее выпущенных версий $v_m$, представляемых в виде набора, координатам которого сопоставляется множество характеристик $F$. Схематическое отображение модели представления программного обеспечения изображено на рисунке 6.

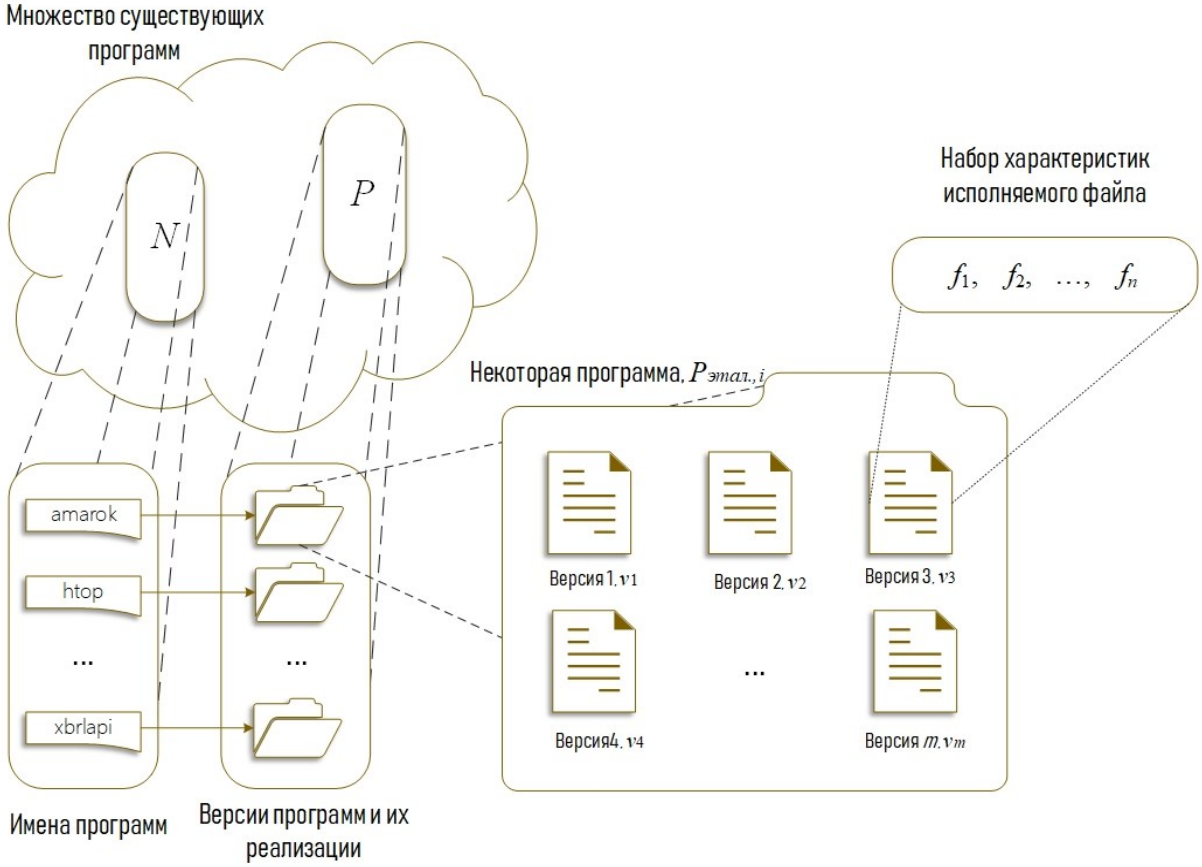


*Рисунок 6 – Модель представления программного обеспечения*

### *Модель представления исполняемого файла*

Исходя из понятий векторной модели, каждый исполняемый файл можно представить в качестве точки $n$-мерного пространства действительных чисел, т.е. вектора, где $n$ – количество градаций признака, являющееся одинаковым для всех рассматриваемых исполняемых файлов.

Если рассмотреть $n$-мерное признаковое пространство $F = (f_1, f_2, \ldots, f_n)$, где $i = 1, 2, \ldots, n$ – число градаций признака, тогда каждый объект $o$ (файл $v_m$ или $e_j$) в данном пространстве будет отображаться точкой с координатами $o = (f_1(o), f_2(o), \ldots, f_n(o))$, а каждый класс объектов, т.е. имя программы $P_{\text{этал.},i}$ – множеством таких точек.

Любой исполняемый файл может быть представлен как совокупность некоторых количественных значений. Ранее было выделено, что наиболее подходящим способом сбора характеристик ПО в данном исследовании является статический сбор частотных характеристик. Нескольким версиям одной и той же программы свойственны определенные характерные особенности. Измерение частот этих особенностей (градаций признака) позволяет нам представить исполняемый файл как последовательность частот характеристик и перенести его вектор в пространство. Исполняемый файл может быть представлен как точка в пространстве, координатами которой являются частоты его градаций признака.

Считая, что идентифицируемый файл обладает некоторыми свойственными ему характеристиками, можно сопоставить ему определенные характеристики конкретных программ. Идентификация файла происходит путем применения выбранного алгоритма сравнения к двум векторам значений признака объектов: $o$ – идентифицируемого и $o'$ – эталонного.

Представление программы в том или ином виде (бинарном, ассемблерном, на уровне инструкций высокоуровневого языка программирования, структуры формата файла и других) обладает некоторыми особенностями.

Очевидно, что наиболее информативным представлением программы будет представление на языке ее исходного кода, однако процесс декомпиляции не всегда позволяет точно его воссоздать и зависит от объема известной о программе информации. В свою очередь, наименее информативным является представление программы в бинарном виде. Таким образом, наиболее информативным и достаточно точным является представление программы в дизассемблированном коде. При этом, выбирая количество анализируемых ассемблерных команд, мы можем изменять степень свободы и тем самым влиять на точность идентификации при разработке методики идентификации исполняемых файлов.

### *Анализ особенностей ассемблерного кода программного обеспечения*

*Уровень абстракции* – наличие смысловых конструкций, упрощающих понимание структуры данных и операций над ними для человека. Высокий уровень абстракции используется в высокоуровневых языках программирования для описания структур данных и операций над ними, описание которых на машинном коде слишком длинно и сложно для понимания человеком. В нем присутствует широкий выбор команд, а имена переменных задаются программистом, в дальнейшем такой программный код компилируется и переводится на низкоуровневый язык (например, ассемблер или байт-код). Чем выше уровень абстракции, тем разнообразнее можно реализовать некоторую функцию в программе и необходимее принимать больше усилий в детальном исследовании всех процедур и данных программы. В то же время представление исполняемого файла на низкоуровневом языке программирования значительно сокращает число возможных используемых значений признака, в том числе таких, чьи функциональные назначения носят схожий характер. Такой код становится легче анализировать и автоматизировать подход к идентификации, не привлекая к работе человека;

*Диапазон признака*. Количество вариаций некоторой характеристики файла носит ограниченный характер. Известно, что степень свободы в статистике – число значений, которые могут варьироваться, и она играет важную роль в количестве информации, носимой признаковым пространством. Так, при рассмотрении признака, имеющего три градации (степень свободы равна двум), количество информации, которое будет носить построенная на нем сигнатура, будет ниже, чем при использовании признака с десятью градациями (степень свободы равна девяти);

*Уникальность и информативность значения признака*. Значение признака может не носить никакой информативной составляющей, как например, значение байта «00», присутствие которого во всех файлах носит скорее заполняющий характер, нежели информационный. Очевидно, что некоторые значения признака будут обладать более существенными идентифицирующими способностями, таковыми обычно являются редко встречающиеся градации признака, они и обладают наибольшей информативностью. Однако в рамках разработанной методики идентификации следовало учитывать разнообразность, присущую исполняемым файлам, участвующим в процессе аудита электронных носителей информации. Программы отличаются не только функциональной направленностью, но и объемом, что существенно влияет на вид формируемых сигнатур. Если в качестве признака выбрать редко встречаемую характеристику файла, то формируемая сигнатура будет содержать большое число нулей, не несущих никакой информации. При этом есть большая вероятность, что для исполняемых файлов небольшого

объема с сильно отличающейся функциональностью выбранный признак и вовсе будет давать нулевую сигнатуру, анализ и сравнение которой будет невозможен.

Автором пособия предлагается производить отбор признаков, которые обладают наиболее высокой различающей способностью и обеспечивают достаточную частоту встречаемости для формирования ненулевых сигнатур исполняемых файлов.

Во второй части учебно-методического пособия будут рассмотрены несколько подходов к идентификации ELF-файлов, каждый из которых обладает своими ограничениями, которые следует учитывать еще на этапе формирования сигнатур файлов. Здесь же рассмотрим выбор признакового пространства.

Среди всего многообразия ассемблерных команд были отобраны 118, которые присущи файлам обучающей выборки, сформированной на основе программ из официальных репозиториев Linux. Однако, в ходе работы над исследованием [1], было решено проанализировать информативность этих команд и произвести отбор идентификационных признаков с наибольшей информативностью.

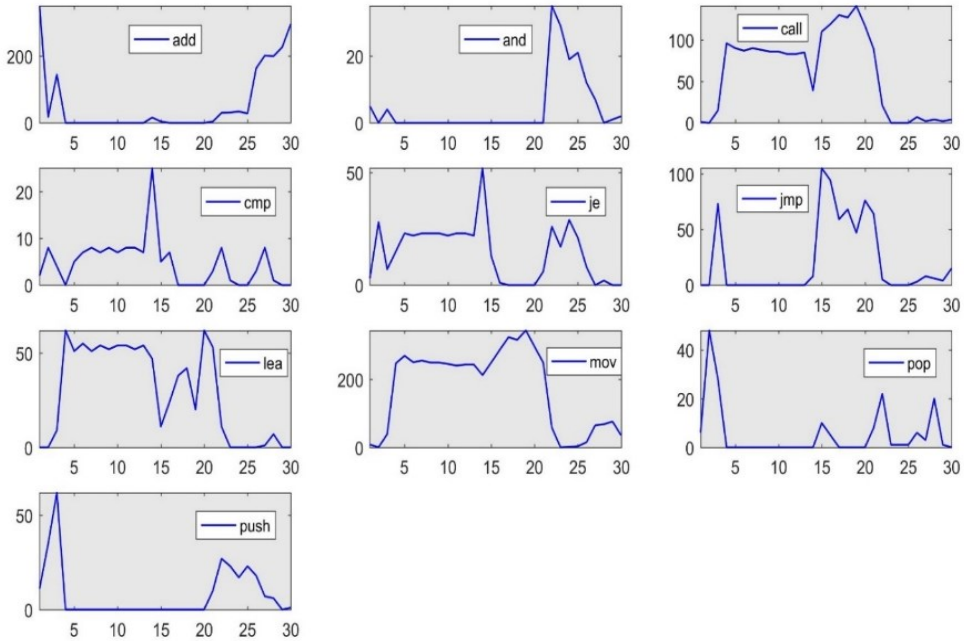


*Рисунок 7 – Частотные распределения ассемблерных команд*

При помощи расчета информативности по методу Шеннона было выделено 10 ассемблерных команд. Посмотрим, как же различается их расположение в дизассемблированном коде программы. Так, на рисунке 7 представлены графики распределений частот выбранных ассемблерных команд для одной версии программы *Amarok* (программа для проигрывания аудиофайлов). На оси абсцисс отмечены номера интервалов (от 1 до 30) разбиения всего тела ассемблерного кода программы, а на оси ординат – частоты встречаемости той или иной ассемблерной команды.

Из графиков рисунка 7 видно, что частотное распределение каждой из ассемблерных команд на интервалах разбиения исполняемого файла весьма различно. Это различие выражается не только в форме получаемого распределения, но и порядке частоты встречаемости признака: одни команды располагаются в основном в середине программы, другие в начале и в конце.

*Контрольные вопросы:*

1. Какова одна из возможных моделей представления программного обеспечения? Исполняемого файла?
2. Почему важным является процесс выбора признакового пространства?
3. Какими свойствами может обладать признак?
4. Придумайте свой вариант признакового пространства, используемого для идентификации программного обеспечения.

# Аудит электронных носителей информации для повышения информационной безопасности автоматизированных систем

Опишем, как же методика идентификации исполняемых файлов может быть применена для повышения безопасности информационных систем организации, путем внедрения ее в качестве технической меры по аудиту ПО на электронных носителях информации.

Контроль установленных программ представляет собой важную составляющую в организации защищенного функционирования бизнес-процессов. Методика позволяет осуществлять автоматизированную идентификацию исполняемых файлов формата ELF в соответствии с формируемых архивом легитимных или не легитимных программ.

До применения методики реализация угрозы со стороны злоумышленника может быть произведена, как показано на рисунке 8 [49].

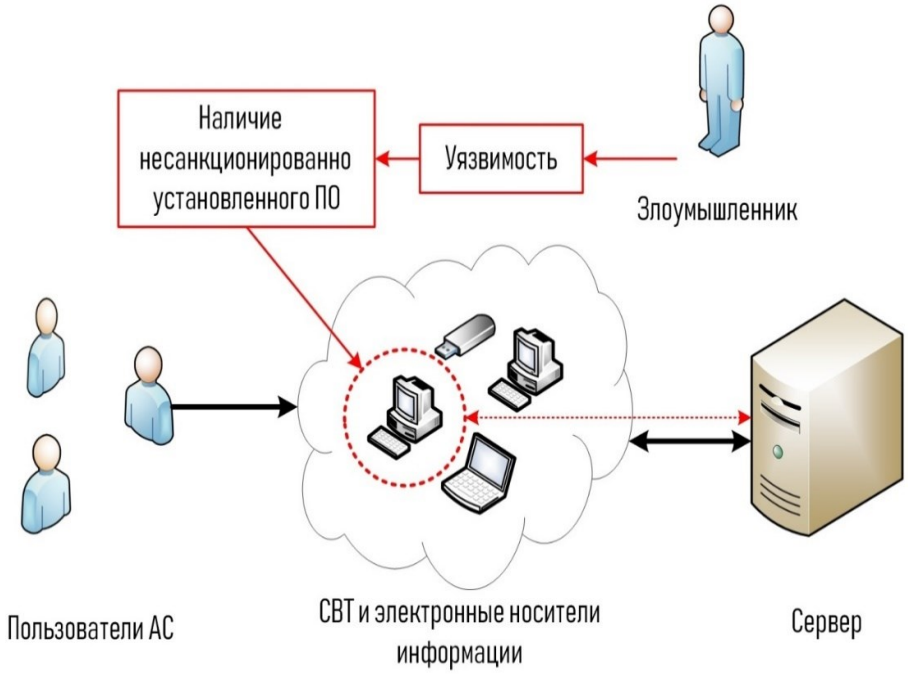


*Рисунок 8 – Схема возможного снижения уровня безопасности системы и воздействие злоумышленника на ИС организации*

Группа пользователей АС взаимодействует со средством вычислительной техники (СВТ) и другими электронными носителями информации, присутствующими в организации и соединенными с главным сервером (или серверами, хранящими и обрабатывающими защищаемую информацию). В условиях соблюдения установленной политики безопасности службе защиты информации известны присущие ИС уязвимости и сформирована стратегия по минимизации возможных рисков

при инцидентах ИБ. Однако, при появлении несанкционированно установленного ПО, присутствует вероятность возникновения новой уязвимости в системе, существование которой не будет известно службе безопасности до тех пор, пока не будет проведен повторный мониторинг АС для выявления уязвимостей.

В этот промежуток времени злоумышленник, обладая необходимыми знаниями и средствами, способен произвести атаку на сервер организации путем эксплуатации возникшей уязвимости из-за несанкционированной установки ПО.

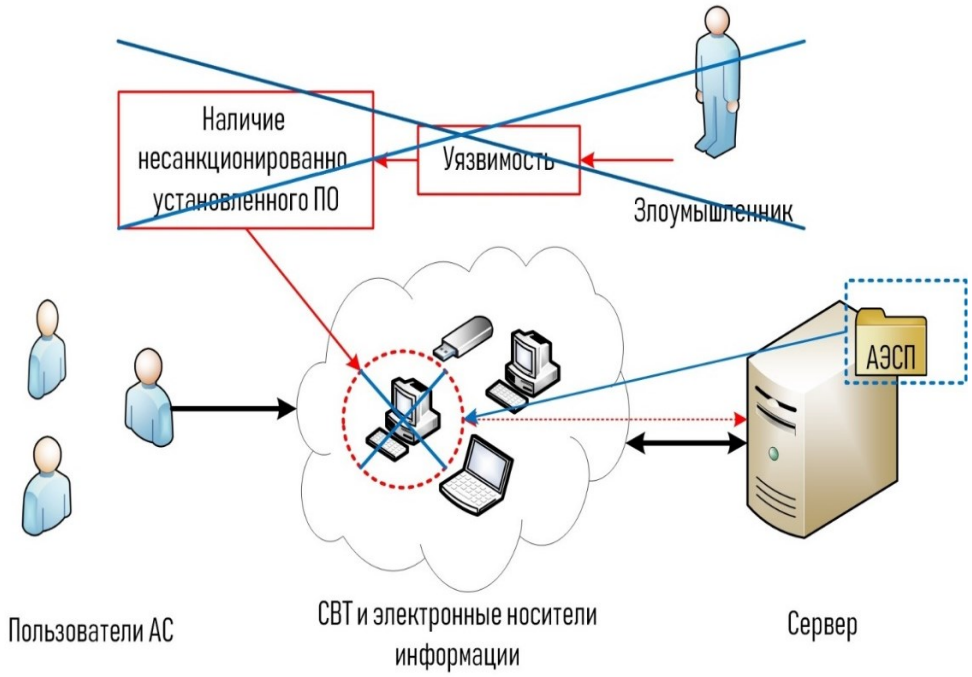


*Рисунок 9 – Схема возможного снижения уровня безопасности системы и воздействие злоумышленника на ИС организации, включающая дополнительную меру по аудиту ПО*

Применение разработанной методики в ИС организации может предотвратить описанный сценарий путем проведения автоматизированного аудита установленного ПО на СВТ и электронных носителях информации (рисунок 9).

Так, располагая архивом эталонных сигнатур программ (АЭСП) (сигнатур, с которыми будет происходить сравнение сигнатуры идентифицируемого исполняемого файла) на стороне сервера и проводя сравнения сигнатур идентифицируемых исполняемых файлов всех присутствующих программ с каждого СВТ и подключаемых к ним съемных электронных носителей информации, можно в короткие сроки уведомить службу ИБ о нарушении правил безопасности и принять своевременные защитные меры.

Практическая реализация заключается в выявлении факта нарушения установленных организационных мер о запрете на установку определенных

программ путем распознавания установленного исполняемого файла как той или иной программы в АЭСП или не входящей в данный архив.

Можно выделить также смежные задачи:

– выявление недобросовестных пользователей, многократно устанавливающих запрещенное ПО;

– выявление пользователей, пытающихся обойти меры по обеспечению ИБ;

– анализ конкретной программы, на основе версий других, схожих по функционалу программ, для выявления степени принадлежности их области назначения.

*Контрольные вопросы:*

1. Опишите возможную схему возникновения уязвимости в автоматизированной системе при несанкционированной установке программы?
2. Какие организационные и технические меры в рамках аудита программного обеспечения обеспечат снижение риска информационной безопасности?
3. Какие смежные задачи можно выделить, решение которых может быть обеспечено идентификацией невредоносных программ?
4. В чем заключается отличие идентификации вредоносных программ и невредоносных?
5. С какими трудностями можно столкнуться при анализе и идентификации программного обеспечения?

# Список сокращений

| | | |
|---|---|---|
| CRC | – | циклический избыточный код (cyclic redundancy check) |
| ELF | – | формат исполнимых и компонуемых файлов (Executable and Linkable Format) |
| HAM | – | управление аппаратными активами (hardware asset management) |
| ITAM | – | управление ИТ-активами (IT asset management) |
| SAM | – | управление программными активами (software asset management) |
| АС | – | автоматизированная система |
| АЭСП | – | архив эталонных сигнатур программ |
| ЗИ | – | защита информации |
| ИБ | – | информационная безопасность |
| ИС | – | информационная система |
| ПО | – | программное обеспечение |
| УК РФ | – | уголовный кодекс российской федерации |
| ЭВМ | – | электронно-вычислительная машина |
| ЭСП | – | эталонная сигнатура программы |

# Список литературы


1. Салахутдинова К.И. Методика идентификации исполняемых файлов на основе статического анализа характеристик дизассемблированного кода программ // Диссертация. СПИИРАН Д 002.199.01. 2020г. 163с.
2. Tool Interface Standards (TIS). Executable and Linking Format (ELF) Specification Version 1.2. TIS Commit. 1995. 106 с.
3. ГОСТ Р ИСО/МЭК 27002-2012 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Свод норм и правил менеджмента информационной безопасности. М.: Стандартинформ, 2012.
4. ГОСТ Р ИСО/МЭК 27001-2006 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Требования. М.: Стандартинформ, 2006.
5. ГОСТ Р 50922-2006. Защита информации. Основные термины и определения. М.: Стандартинформ, 2006.
6. ФЗ от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации (с изменениями на 18 декабря 2018 года)». Собрание законодательства Российской Федерации, N 31 (ч.I), 31.07.2006, ст.3448, 2006.
7. Закон РФ от 21 июля 1993 г. №5485-1 «О государственной тайне (с изменениями на 29 июля 2018 года)». Собрание законодательства Российской Федерации, N 41, 13.10.1997, ст.4673, 1993.
8. Указ Президента РФ от 6 марта 1997 г. №188 «Об утверждении перечня сведений конфиденциального характера (с изменениями на 13 июля 2015 года)». Собрание законодательства Российской Федерации, N 10, 10.03.97, ст.1127, 1997.
9. ФЗ от 20 февраля 1995 г. № 24-ФЗ «Об информации, информатизации и защите информации (с изменениями на 10 января 2003 года) (утратил силу с 09.08.2006 на основании Федерального закона от 27.07.2006 N 149-ФЗ)». Собрание законодательства Российской Федерации N 8, 20.02.95, ст.609. ст.2 с.
10. Петренко С.А., Курбатов В.А. Политики безопасности компании при работе в Интернет. 2011. 396 с.
11. Boukhtouta A. и др. Graph-theoretic characterization of cyber-threat infrastructures // Digit. Investig. 2015. Т. 14, № 1. С. 3–15.
12. Бондарев В. Введение в информационную безопасность автоматизированных систем. 2018. 252 с.
13. Software Management: Security Imperative, Business Opportunity [Электронный ресурс]. 2018. С. 24. URL: https://gss.bsa.org/wp-content/uploads/2018/05/2018_BSA_GSS_Report_en.pdf. Режим доступа: свободный, дата обращения: 18.03.2018.
14. Отчет о видах наказания по наиболее тяжкому преступлению (без учета сложения) [Электронный ресурс]. 2017. URL: http://www.cdep.ru/index.php?id=79&item=4572. Режим доступа: свободный, дата обращения: 11.12.2018.
15. Базовая модель угроз безопасности персональных данных при их обработке в информационных системах персональных данных (выписка). ФСТЭК России, 2008 год. 2008.

16. Быкова В.В., Катаева А.В. Методы и средства анализа информативности признаков при обработке медицинских данных. // Журнал Программные продукты и системы / Softw. Syst. 2016. Т. 2, № 114. С. 172–178.
17. Hastie T., Tibshirani R., Friedman J. The elements of statistical learning: Data Mining, Inference, and Prediction. // Springer. 2017. 745 с.
18. Еремеев М.А., Кравчук А.В. Анализ методов распознавания вредоносных программ. // Журнал Вопросы защиты информации. 2014. Т. 3. С. 44–51.
19. Giesen D. Philosophy and practical implementation of static analyzer tools // Softw. Dev. J. 1998. № 7. С. 12–31.
20. Christopher C.N. Evaluating Static Analysis Frameworks. 2003. 213с.
21. Seo K. и др. Detecting Similar Files Based on Hash and Statistical Analysis for Digital Forensic Investigation // Digital Forensic Investigation. In: Computer Science and its Applications (CSA 2009). 2009. С. 1–6.
22. Breitinger F. B.H. Similarity Preserving Hashing: Eligible Properties and a New Algorithm MRSH-v2 // Digital Forensics and Cyber Crime. ICDF2C 2012. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. 2012. С. vol 114.
23. Kornblum J.D. Identifying almost identical files using context triggered piecewise hashing // Digit. Investig. 2006. Т. 3. С. 91–97.
24. Long C., Guoyin W. An Efficient Piecewise Hashing Method for Computer Forensics // First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008). 2008. С. 635–638.
25. Baier H., Frank B. Security Aspects of Piecewise Hashing in Computer Forensics // 2011 Sixth International Conference on IT Security Incident Management and IT Forensics. С. 21–36.
26. Цифровые подписи в Windows 7 [Электронный ресурс]. 2012. URL: http://windowsnotes.ru/windows-7/cifrovye-podpisi-v-windows-7. Режим доступа: свободный, дата обращения: 23.04.2018.
27. Niemela J. It's Signed, therefore it's Clean, right? [Электронный ресурс] // Protecting the irreplaceable, f-secure.com. 2010. С. 1–30. URL: https://www.f-secure.com/weblog/archives/Jarno_Niemela_its_signed.pdf. Режим доступа: свободный, дата обращения: 23.04.2018.
28. Willems C., Holz T., Freiling F. Toward automated dynamic malware analysis using cwsandbox // 2007 IEEE Symposium on Security and Privacy (S&P 2007). 2007. С. 32–39.
29. Bayer U., Kruegel C., Kirda E. Ttanalyze: A tool for analyzing malware // 15th European Institute for Computer Antivirus Research (EICAR 2006). 2006. С. 180– 192.
30. Song D. и др. Bitblaze: A new approach to computer security via binary analysis // 4th International Conference on Information Systems Security. 2008. С. 1–25.
31. AVG [Электронный ресурс]. URL: https://www.avg.com/en-ww/avg-andnorman-business-products?_ga=2.1955633.1678056726.1552380509-211382991.1552380509.
32. Corporation S. Advanced Threat Protection [Электронный ресурс]. URL: https://www.symantec.com/products/threat-protection.
33. Lin C.H., Pao H.K., Liao J.W. Efficient dynamic malware analysis using virtual time control mechanics // Comput. Secur. 2018. Т. 73. С. 359–373.
34. Egele M. и др. A Survey on Automated Dynamic Malware-Analysis Techniques and Tools // ACM Comput. Surv. 2012. Т. 44, № 2. С. 42.
35. Gray-Donald T.A., Price M.W. Date and time simulation for time-sensitive 138 applications: пат. US 8,418,151 B2 USA. USA, 2013. С. 9.

36. Bulazel A., Bülent Y. A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion: Pc, mobile, and web. // the 1st Reversing and Offensiveoriented Trends Symposium. 2017. С. 1–21.
37. Windows Privilege Escalation Fundamentals [Электронный ресурс]. 2014. URL: http://www.fuzzysecurity.com/tutorials/16.html.
38. Schultz M.G. и др. Data mining methods for detection of new malicious executables // Proceedings of the IEEE Symposium on Security and Privacy. Los Alamitos, CA, 2001. С. 38–49.
39. Вартанов С.П., Герасимов А.Ю. Динамический анализ программ с целью поиска ошибок и уязвимостей при помощи целенаправленной генерации входных данных // Труды ИСП РАН. 2014. Т. 26, № 1. С. 375–394.
40. Мельников П.В., Горюнов М.Н., Анисимов Д.В. Подход к проведению динамического анализа исходных текстов программ // Вопросы кибербезопасности. 2016. Т. 3, № 16. С. 33–39.
41. Klosterboer L. Implementing ITIL configuration management. 2007. 264 с.
42. England R. Owning ITIL. 2009. 178 с.
43. Bird J. DevOpsSec. 2016.
44. Samanage. Samanage [Электронный ресурс]. 2019. URL: https://www.samanage.com (дата обращения: 28.03.2019).
45. Kaspersky. Kaspersky Systems Management [Электронный ресурс]. 2019. URL: https://www.kaspersky.ru (дата обращения: 28.03.2019).
46. Microsoft. Microsoft Assessment and Planning Toolkit [Электронный ресурс]. 2018. URL: https://www.microsoft.com/en-us/Download/details.aspx?id=7826 (дата обращения: 28.03.2019).
47. FinalWire. AIDA64 [Электронный ресурс]. URL: https://www.aida64.com (дата обращения: 28.03.2019).
48. Lansweeper. Lansweeper [Электронный ресурс]. 2019. URL: https://www.lansweeper.com (дата обращения: 28.03.2019).
49. Салахутдинова К.И., Лебедев И.С., Кривцова И.Е., Анисимов А.С. Идентификации программного обеспечения в задаче аудита электронных носителей информации // Авиакосмическое приборостроение. 2019. № 9. С. 20–28.

Салахутдинова Ксения Иркиновна

**Методы идентификации открытого программного обеспечения.**
**Часть I**
**Учебно-методическое пособие**

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

**Редакционно-издательский отдел**
**Университета ИТМО**
197101, Санкт-Петербург, Кронверский пр., 49