

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский технический университет связи и информатики»

Разрешаю
допустить к защите
Зав. кафедрой

13 VI 2023 г.

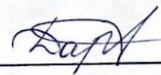
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ

РАБОТА

НА ТЕМУ

Разработка программного обеспечения для выявления
вредоносных программ в операционной системе Linux

Студент: Никольская Дарина Игоревна



Научный руководитель: Симонян А. Г.



Москва 2023 г

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**
Ордена Трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра _____ «Информационная безопасность»
(название полностью)
«Утверждаю» _____
Зав. кафедрой _____
« 06 » 05 2023 г.

ЗАДАНИЕ
на выпускную квалификационную работу

Студенту _____ Никольской Дарины Игоревны _____ гр. _____ БП31901

Направление (специальность) _____ 11.03.02 Инфокоммуникационные технологии и системы
связи _____

Форма выполнения выпускной квалификационной работы _____
бакалаврская работа
(Дипломный проект, дипломная работа, магистерская диссертация, бакалаврская работа)

Тема выпускной квалификационной работы _____ «Разработка программного обеспечения для
выявления вредоносных программ в операционной системе Linux» _____

Утверждена приказом ректора № _____ от _____ 20 _____ г.

1. Исходные данные
Операционная система – Linux
Используемое ПО – PyCharm

2. Содержание расчетно-пояснительной записки
(перечень подлежащих разработке вопросов)

Введение

Глава 1. Принципы работы ПО для выявления вредоносных приложений в ОС Linux.

Глава 2. Разработка алгоритмов работы приложения для выявления вредоносного ПО.

Глава 3. Программная реализация средства для выявления вредоносного ПО на основе разработанных алгоритмов в ОС Linux.

Глава 4. Обнаружение вредоносных программ в ОС Linux, с помощью разработанного программного обеспечения.

Заключение

Объем работы в % и сроки
выполнения по разделам

5% 05.04.2023

20% 25.04.2023

25% 05.05.2023

25% 15.05.2023

20% 25.05.2023

5% 31.05.2023

3. Вопросы конструктивных разработок

4. Разработка вопросов по экологии и безопасности жизнедеятельности

5. Технико-экономическое обоснование (подлежащее расчету)

6. Перечень графического материала (с точным указанием обязательных чертежей)

1. Актуальность применения ПО для выявления вредоносного ПО.
2. Принципы работы приложения для выявления вредоносного ПО.
3. Блок-схема алгоритма работы приложения для выявления вредоносного ПО в ОС Linux.
4. Интерфейс разработанного ПО.
5. Результаты работы разработанного приложения для выявления вредоносного ПО в ОС Linux.
6. Заключение.

7. Консультанты по ВКР (с указанием относящихся к ним разделов проекта):

(подпись)

(ФИО)

(подпись)

(ФИО)

8. Срок сдачи студентом законченной ВКР: 31.05.2023г.

Дата выдачи задания: 01.04.2023г.

Руководитель

(подпись)

Симонян Айрапет Генрикович

(ФИО)

почасовая

(штатная или почасовая)

нагрузка

Задание принял к исполнению

(подпись студента)

Примечание: Настоящее задание прилагается к законченной ВКР

Отзыв
руководителя на выпускную квалификационную работу студента
Никольской Дарины Игоревны
на тему: «Разработка программного обеспечения для выявления
вредоносных программ в операционной системе Linux»

Актуальность темы выпускной квалификационной работы связана с необходимостью обеспечения безопасности персональных компьютеров в связи со значительным ростом количества атак с помощью вредоносных приложений. В том числе на компьютеры работающие с использованием ОС Linux. Исходя из вышесказанного, бакалаврская работа студента Никольской Д.И. является актуальной.

При выполнении ВКР Никольская Д.И. проделала значительный объем самостоятельной работы по анализу и сбору данных, продемонстрировала навыки в работе со средой разработки и способности в решении поставленных задач.

В первом разделе работы выполнен анализ методов выявления вредоносных программ в ОС Linux. Рассмотрены их особенности, достоинства и недостатки при использовании. Во втором разделе представлены разработанные алгоритмы работы ПО и их блок-схемы для выявления вредоносных программ в ОС Linux. В третьем разделе представлена программная реализация приложения на основе разработанных алгоритмов. В четвертой главе представлены результаты работы разработанного ПО. Показаны результаты тестирования разработанного приложения. В заключении даны выводы на основе выполненной работы.

Материал в работе логически структурирован и достаточно хорошо проиллюстрирован.

При работе над ВКР Никольская Д.И. проявила себя как инициативный, трудолюбивый и дисциплинированный специалист, способный самостоятельно решать поставленные задачи на высоком профессиональном уровне, а полученный объем знаний помог решить ей сложную задачу грамотно и квалифицированно. Отлично ориентируется в технической литературе. При выполнении работы все поставленные цели достигнуты, отступлений от графика работы не было.

Считаю, что стандартные компетенции у Никольской Дарины Игоревны сформированы на высоком уровне, а Никольская Дарина Игоревна – заслуживает присвоения степени «бакалавр» по направлению 11.03.02 – «Инфокоммуникационные технологии и системы связи».

Руководитель



к.т.н., доцент Симонян А.Г.

Аннотация

Бакалаврская работа на тему: "Разработка программного обеспечения для выявления вредоносных программ в операционной системе Linux"

Работа содержит 55 страниц текста, включает 24 рисунка и список из 25 наименований информационных источников.

Целью данной работы является разработка программного обеспечения для выявления вредоносных приложений в операционной системе Linux.

В работе рассмотрены методы выявления вредоносного программного обеспечения, а также особенности программного обеспечения для выявления вредоносных программ в ОС Linux. Представлены разработаны алгоритмы работы приложения для выявления вредоносного ПО в ОС Linux, а также их программная реализация. Выполнено тестирование разработанного ПО.

Оглавление

Введение	8
1. Принципы работы ПО для выявления вредоносных приложений в ОС Linux. 9	
1.1 Актуальность применения защиты от вредоносного ПО	9
1.2 Методы выявления вредоносного ПО.	11
1.3 Особенности работы ПО для выявления вредоносных программ в ОС Linux.	15
1.4 Выводы по первой главе	17
2. Разработка алгоритмов работы приложения для выявления вредоносного ПО.	19
2.1 Обобщенный алгоритм работы приложения для выявления вредоносного ПО.	19
2.2 Алгоритм работы модуля выявления вредоносных приложений.	20
2.3 Алгоритм работы модуля выявления изменений в файле.	22
2.4 Алгоритм работы модуля карантин	24
2.5 Выводы по второй главе.....	26
3. Программная реализация средства для выявления вредоносного ПО на основе разработанных алгоритмов в ОС Linux.	27
3.1 Выбор языка и среды разработки для выявления вредоносного ПО.....	27
3.2 Разработка интерфейса ПО для выявления вредоносных приложений.	27
3.3 Программная реализация модуля поиска сигнатур вредоносных приложений.....	29
3.4 Программная реализация модуля «карантин».	30
3.5 Программная реализация модуля выявления изменений в файле.	31
3.6 Выводы по третьей главе	31
4. Обнаружение вредоносных программ в ОС Linux, с помощью разработанного программного обеспечения.	33
4.1 Выявление вредоносных приложений с помощью модуля поиска сигнатур	33

4.2 Обнаружение вредоносных приложений с помощью модуля выявления изменений в файле.....	36
4.3 Выводы по четвертой главе	43
Заключение.....	45
Список используемых источников	47
Приложение.....	50

Введение

В современном мире все больше компьютерных систем становятся доступны из глобальной сети, вследствие чего пользователи сталкиваются с возрастающим числом различных угроз – сетевыми атаками, несанкционированным доступом, вредоносным программным обеспечением (ВПО).

Вредоносное ПО – программный код, который может использоваться для кражи данных, обхода средств контроля доступа, причинения вреда, компрометации данных и других действий, наносящих ущерб. Примеры вредоносного ПО: вирусы, загрузочные вирусы, скрипт-вирусы, программы для кражи паролей, трояны, шпионские программы, и другие.

В настоящее время операционная система (ОС) Linux используется не только на личных компьютерах, но также в офисах, медицинских учреждениях и т.д. С ростом ее востребованности актуальным становится вопрос связанный с обеспечением ее защиты. При этом вредоносные программы все чаще осуществляют целевые атаки на ОС Linux, распространяются с помощью запускаемых файлов, скриптов, получаемых по почте или передаваемых с помощью переносных носителей.

Целью данной работы является разработка программного обеспечения для выявления вредоносных приложений в операционной системе Linux. Для достижения данной цели необходимо решить следующие задачи:

- Выполнить анализ существующих методов выявления вредоносного ПО в ОС Linux;
- Разработать алгоритмы работы ПО для выявления вредоносных приложений в ОС Linux;
- Программно реализовать ПО для выявления вредоносных программ в ОС Linux;
- Выявление вредоносного ПО в ОС Linux с помощью разработанного приложения.

1. Принципы работы ПО для выявления вредоносных приложений в ОС Linux.

1.1 Актуальность применения защиты от вредоносного ПО

По мере появления огромного числа вредоносного ПО, решающее значение для специалистов имеют способы реагирования на него. Анализ системы на наличие вредоносных программ стал обязательным в современном мире для обеспечения безопасности системы. Такой анализ требует сбалансированного подхода для решения проблем с безопасностью.

Статистика атак, с использованием вредоносных программ в 2022 году, показывает, что они являются существенной проблемой не только для физических лиц, но и для организаций [1]. На рисунке 1.1 представлена статистика, подвергшихся атакам категорий пользователей в 2022 году.

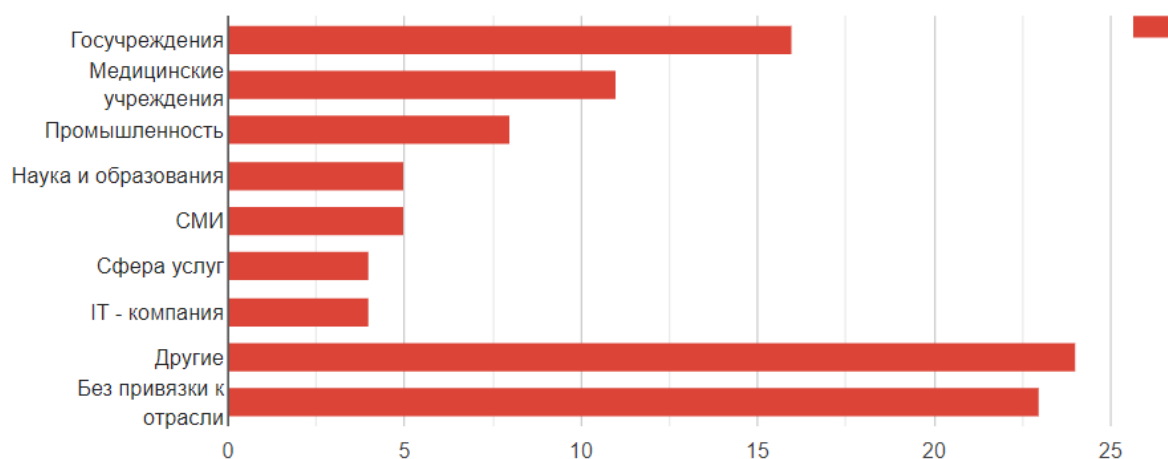


Рисунок 1.1- Категории пользователей, подвергшихся атакам.

Понятие вредоносное ПО включает в себя различные типы вредоносных программ, такие как вирусы, черви и т.д. При выполнении анализа, на предмет наличия вредоносного ПО в системе часто встречаются различные их типы, некоторые из которых классифицируются в зависимости от функциональных возможностей и векторов атаки. На рисунке 1.2 представлены типы вредоносного ПО наиболее распространённые в 2022 году [2].



Рисунок 1.2- Типы вредоносного ПО.

Большинство атак не поддается огласки, поэтому подсчитать точное число реализованных угроз невозможно даже для организаций, занимающихся расследованием инцидентов и анализом действий хакерских групп.

Существует множество причин, почему защита от вредоносного ПО остается необходимой, а ее применение актуальной. Например, это могут быть следующие причины:

- 1.Рост числа вредоносных программ. Количество вредоносных программ, растет с каждым днем. Киберпреступники используют различные методы, чтобы проникнуть на компьютер, украсть данные и использовать их для своих целей.

- 2.Угроза для безопасности данных. Это может быть конфиденциальная информация, финансовые данные, личная информация и многое другое. Киберпреступники могут использовать такие данные для вымогательства, кражи, шантажа, для взлома систем и другие.

- 3.Угроза для финансовой стабильности. Вирусы и другие формы вредоносного ПО могут привести к финансовым потерям для компании или для отдельного пользователя. Например, если киберпреступники получают доступ к кредитным картам или банковским счетам, это может привести к значительным финансовым потерям.

4.Необходимость защиты от новых видов угроз. Киберпреступники постоянно разрабатывают новые методы и технологии, чтобы проникать в компьютеры и сети. Необходимо обновлять используемые ПО для выявления вредоносных программ и дополнительные защитные средства, чтобы бы система была защищена от новых вирусов и других угроз.

1.2 Методы выявления вредоносного ПО.

Наиболее распространённым способом обнаружения угроз в ОС Linux, является применение сигнатурного метода, в виду низкого уровня ложных срабатываний. В нем описывается каждая атака или угроза отдельной моделью или сигнатурой, в качестве которой могут применяться строка символов, семантическое выражение, формальная математическая модель и т.д. Применение сигнатурного метода позволяет защитить от вирусной или хакерской атаки, когда сигнатура уже известна и внесена в базу данных [8].

Задачу выделения сигнатур, как правило, решают в компаниях, разработчиках антивирусных приложений, для которых выделяются характерные черты и фрагменты программного кода. Как правило - в наиболее простых случаях могут применяться автоматизированные средства выделения сигнатур. Например, в случае несложных по структуре программ, у которых отсутствует задача заражения других программ, и их код целиком является вредоносным ПО.

Сигнатурный метод используется в большинстве ПО осуществляющего выявление вредоносных программ. Когда антивирусный сканер проверяет файл в ОС Linux, то производится сравнение содержимого файла с сигнатурной базой известных вирусов. Если находится совпадение, ПО для выявления вредоносных программ обозначает этот файл как зараженный и выполняет соответствующие действия [3].

К достоинствам сигнатурного метода можно отнести следующее:

- высокая точность обнаружения вредоносного ПО, так как анализируются конкретные сигнатуры;

- экономия времени, так как не требуется запускать вредоносную программу;

Важным, но отрицательным свойством такого метода является то, что для создания сигнатуры необходимо иметь образец вируса. Следовательно, сигнатурный метод непригоден для защиты от новых вирусов, т.к. до тех пор, пока вирус не попал на анализ к экспертам, создать его сигнатуру невозможно. Также ПО для выявления вредоносных программ, использующее данный метод может пропустить зараженный файл, если вредоносная программа изменяет свою сигнатуру. С момента появления вредоносной программы в сети Интернет до выпуска первых сигнатур может пройти от нескольких часов до нескольких дней. Все это время вредоносное ПО способно заражать компьютеры почти беспрепятственно и ПО для выявления вредоносных программ, использующие только сигнатурный метод обнаружения не смогут выявлять их.

Поэтому для защиты от вредоносного ПО необходимо использовать не только сигнатурный, но и другие методы, например, эвристический метод антивирусной защиты или метод обнаружения изменений.

Эвристический метод выявления вредоносного ПО, не основывается на использовании сигнатур (шаблонов), как это реализовано в сигнатурном методе. С его помощью реализуется защита, основанная на определенных правилах и эвристических алгоритмах, которые позволяют выявлять потенциально вредоносные приложения. Данный способ выявляет вредоносные программы с некоторой вероятностью, поэтому возможны ложные срабатывания.

В работе эвристического анализатора используются две формы эвристики - пассивная и активная.

Пассивная эвристика строится на поиске в программном коде подозрительных признаков (команд), характерных для вредоносных программ. Встретив в коде приложения или файла подобный признак, увеличивается «счетчик подозрительности» данного объекта. При

превышении счетчиком порогового значения объект признается подозрительным. Согласно последним исследованиям специалистов по компьютерной безопасности, даже самые современные анализаторы не способны распознать около 30 % вредоносного кода [2]. Являясь важным инструментом антивирусной защиты, пассивная эвристика не является гарантией 100% защиты, поскольку если какое - либо действие разрешено законной программе, то оно может быть выполнено и вредоносной программой. Также при его использовании высока вероятность ложных срабатываний при наличии в безопасной программе фрагментов кода, исполняющего действия или последовательности, характерные для некоторых угроз. Этим объясняется необходимость использования активной эвристики.

Концепция активной эвристики заключается в создании виртуального компьютера в ядре сканирования, что позволяет сканеру изучить возможные действия программы, если она будет запущена на реальном компьютере [11]. Это позволяет распознать потенциально вредоносные действия, которые не определяются при помощи других способов обнаружения. Метод базируется на выявлении основных и наиболее часто предпринимаемых вредоносных действий, таких как, удаление или создание файла, запись в определенные области системного реестра, открытие порта на прослушивание, перехват данных, вводимых с клавиатуры, рассылка писем и др. Выполнение любого из этих действий по отдельности, часто, не является поводом считать программу вредоносной. Однако в случае, когда программа осуществляет ряд подобных операций можно предположить, что она, по меньшей мере, подозрительна. Такая схема анализа предъявляет более высокие требования к ресурсам персонального компьютера (ПК), так как для анализа приходится использовать безопасное виртуальное пространство, а запуск приложения на компьютере пользователя откладывается на время анализа [4].

Однако у эвристического метода есть и недостатки. При его работе могут возникать ложные срабатывания, т.е. безопасные программы могут

идентифицироваться как опасные или потенциально вредоносные на основе общих правил, это не всегда корректно.

Поведенческий метод выявления вредоносного ПО основан на анализе поведения программы в режиме реального времени. Он позволяет определять потенциально вредоносное поведение программы на основе ее действий, выполняемых в системе [6].

ПО для выявления вредоносных программ, использующие метод обнаружения подозрительного поведения программ, не пытаются идентифицировать известные вирусы, вместо этого они отслеживают поведение запущенных программ. Например, если программа пытается записать какие-то данные в исполняемый файл, ПО для выявления вредоносных программ может отметить этот файл, предупредить пользователя и спросить, что следует сделать.

К основным вредоносным действиям могут быть отнесены:

- удаление файла;
- запись в файл;
- запись в определенные области системного реестра;
- открытие порта на прослушивание;
- перехват данных, вводимых с клавиатуры;
- рассылка писем и др.

В большинстве случаев выполнение каждого такого действия по отдельности не дает повода считать программу вредоносной. Но если программа последовательно выполняет несколько действий последовательность которых занесена в базу подозрительных значит, и программа может быть отнесена как минимум к подозрительным. Поведенческий метод идентифицирует подозрительные действия программы и выдает предупреждение или реагирует на них.

Примеры поведенческих методов:

1. Мониторинг изменений в операционной системе. Различные изменения, направленные на установку вредоносных программ или на получение доступа к компьютеру, могут быть обнаружены и заблокированы.

2. Система контроля процессора. Приложение контролирует работу процессора, чтобы определить подозрительную активность, например, принудительно остановить выполнение программы, если она становится подозрительной. Поведенческий метод позволяет выявить как известные, так и неизвестные типы вирусов, что является неоспоримым достоинством такого подхода к защите.

Поведенческий метод обнаружения вредоносного ПО позволяет справляться с новыми угрозами, которые могут оказаться не выявленными с помощью сигнатурных методов. Однако он может быть менее точным, чем сигнатурный метод, и может приводить к большому количеству ложных срабатываний, если некоторые операции, которые рассматриваются как подозрительные, на самом деле не представляют угрозы.

Для более точного обнаружения ВПО требуется совместное применение нескольких методов. И только комплексный подход к данной проблеме может значительно снизить риск вторжения в информационную систему.

1.3 Особенности работы ПО для выявления вредоносных программ в ОС Linux.

Linux предоставляет множество функций и возможностей, таких как многозадачность, многопользовательский режим, поддержку сети и многое другое. Кроме того, она поставляется с большим количеством программных компонентов, таких как библиотеки, утилиты, среды рабочего стола и т.д., которые позволяют пользователю выполнять большое количество различных задач [18].

В большинстве случаев, Linux более стабильная и безопасная операционная система, чем другие. Она привлекает многих пользователей

своей скоростью работы, надежностью и безопасностью, а также гибкостью и широкими возможностями настройки.

Определить наличие вредоносного ПО в ОС Linux можно по различным признакам. К ним можно отнести следующие:

- Замедление работы компьютера;
- Уменьшение объема оперативной памяти;
- Зависание, перезагрузка или блокировка компьютера;
- Ошибки при работе ОС или прикладных программ;
- Изменение размеров файлов и др.

Наличие одного или нескольких признаков не дает гарантии заражения ОС Linux вредоносным ПО [24].

Антивирусное ПО — это программа для обнаружения, идентификации и устранения вирусов с компьютера или другого устройства.

Различают следующие виды программ для выявления ВПО:

1. Сканер – выявляет ВПО в оперативной памяти, загрузочных областях, на локальных и внешних дисках, а также в системных файлах ОС. Может выполняться по расписанию, по запросу пользователя или при обращении к файлам. Сканеры также можно разделить на две категории - "универсальные" и "специализированные" [5].

Универсальные сканеры рассчитаны на поиск и обезвреживание всех типов вирусов в ОС. Специализированные предназначены для обезвреживания, ограниченного числа вирусов или только одного их класса,

Сканер предоставляет пользователю несколько вариантов на выбор:

- попытаться вылечить, удалив вредоносный код;
- поместить в карантин, чтобы вылечить позже или удалить;
- удалить, если вылечить не удалось;

2. Монитор – отслеживает все манипуляции с файлами в режиме реального времени. Находит и обезвреживает вредоносное ПО до того, как оно успеет инфицировать систему.

Проверка в режиме реального времени обеспечивает непрерывность работы ПО для выявления вредоносных программ. Это реализуется с помощью обязательной проверки всех действий, совершаемых другими программами и самим пользователем в ОС Linux, на предмет вредоносности вне зависимости от их исходного расположения – будь то свой жесткий диск, внешние носители информации, сетевые ресурсы или оперативная память. Также проверке подвергаются все косвенные действия через третьи программы. Такой режим защиты системы от заражения должен быть включен с начала загрузки операционной системы и выключаться в последнюю очередь, при завершении работы ОС Linux.

Во многих приложениях, осуществляющих выявление вредоносных программ среди вспомогательных средств имеется технология – карантин. Карантин - это место для хранения файлов, считающихся подозрительными или вредоносными. Эта область позволяет избежать потери данных в случае некорректных действий ПО для выявления вредоносных программ по отношению к файлам. Так, например, антивирус может ошибочно определить, что файл заражен вредоносным ПО и удалить его.

В таких случаях перед лечением или удалением файлов необходимо сохранить их резервные копии, и если окажется, что файл был удален ошибочно или потеряна важная информация, всегда можно будет выполнить его восстановление.

1.4 Выводы по первой главе

В данном разделе были рассмотрены различные виды вредоносных приложений, методы их обнаружения, а также принцип работы ПО для выявления вредоносных программ на основании проанализированной информации можно сделать выводы:

- Сигнатурный метод является самым точным из способов, благодаря его быстрому реагированию на угрозы и малого количества ложных срабатываний.
- Для повышения безопасности персонального компьютера (ПК) должны применяться все методы антивирусной защиты одновременно дополняя друг друга.
- Не существует ПО для выявления вредоносных программ, гарантирующего 100% защиту, поэтому не смотря на высокий уровень защиты необходимо применять дополнительные средства защиты [11].

Защита от вредоносного ПО является одним из наиболее важных аспектов компьютерной безопасности, который необходимо использовать при работе с компьютером.

У каждого типа ПО для выявления вредоносных программ есть свои достоинства и недостатки. Однако комплексное использование нескольких типов антивирусных программ, основывающихся на различных методах может привести к желаемому результату.

2. Разработка алгоритмов работы приложения для выявления вредоносного ПО.

2.1 Обобщенный алгоритм работы приложения для выявления вредоносного ПО.

Обобщенный алгоритм работы разрабатываемого ПО для поиска вредоносных приложений в ОС Linux функционирует следующим образом. На ПК пользователя, в заданной папке размещаются файлы, содержащие сигнатуры вредоносных программных обеспечений. Во время его работы, пользователь в ручную запускает сканер для проверки файлов на наличие вредоносного содержимого. Следующий модуль выявляет изменения файлов, происходящие на ПК. Проверенные файлы, не удачно прошедшие проверку ПО для выявления вредоносных программ, помещаются в карантин. В случае необходимости пользователь может восстановить их. Пользователь может совершить аналогичные действия с последующим файлом. На рисунке 2.1 представлена обобщенная блок-схема алгоритма работы разрабатываемого ПО для выявления вредоносных программ.

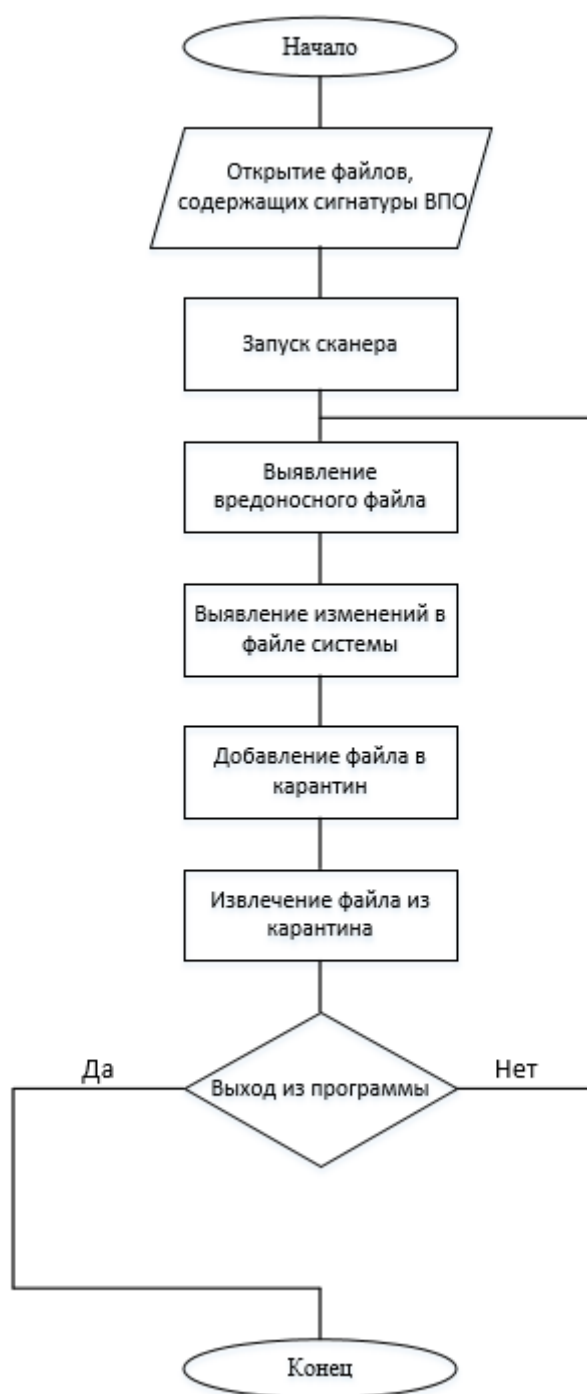


Рисунок 2.1 - Обобщенный алгоритм работы модуля ПО для выявления вредоносных программ.

2.2 Алгоритм работы модуля выявления вредоносных приложений.

После загрузки сканера в ОС Linux, пользователь должен выбрать папку для выявления вредоносного кода. Если папка не выбрана, то пользователь не сможет продолжить работу. Если все файлы в папке прошли проверку и являются безопасным для ПК, программа переходит к их проверке с помощью

следующего модуля. Если нет, то при сканировании каждого файла проходит сравнение его содержимого с сигнатурами, хранящимися в базе данных программы. Если опасность обнаружена, то такой файл автоматически перемещается в папку «карантин». Записи о результатах проверки файла заносятся в журнал событий. Данный алгоритм выполняется до тех пор, пока не будут обнаружены все зараженные файлы. На рисунке 2.2 представлена блок-схема алгоритма работы модуля выявления вредоносного кода.

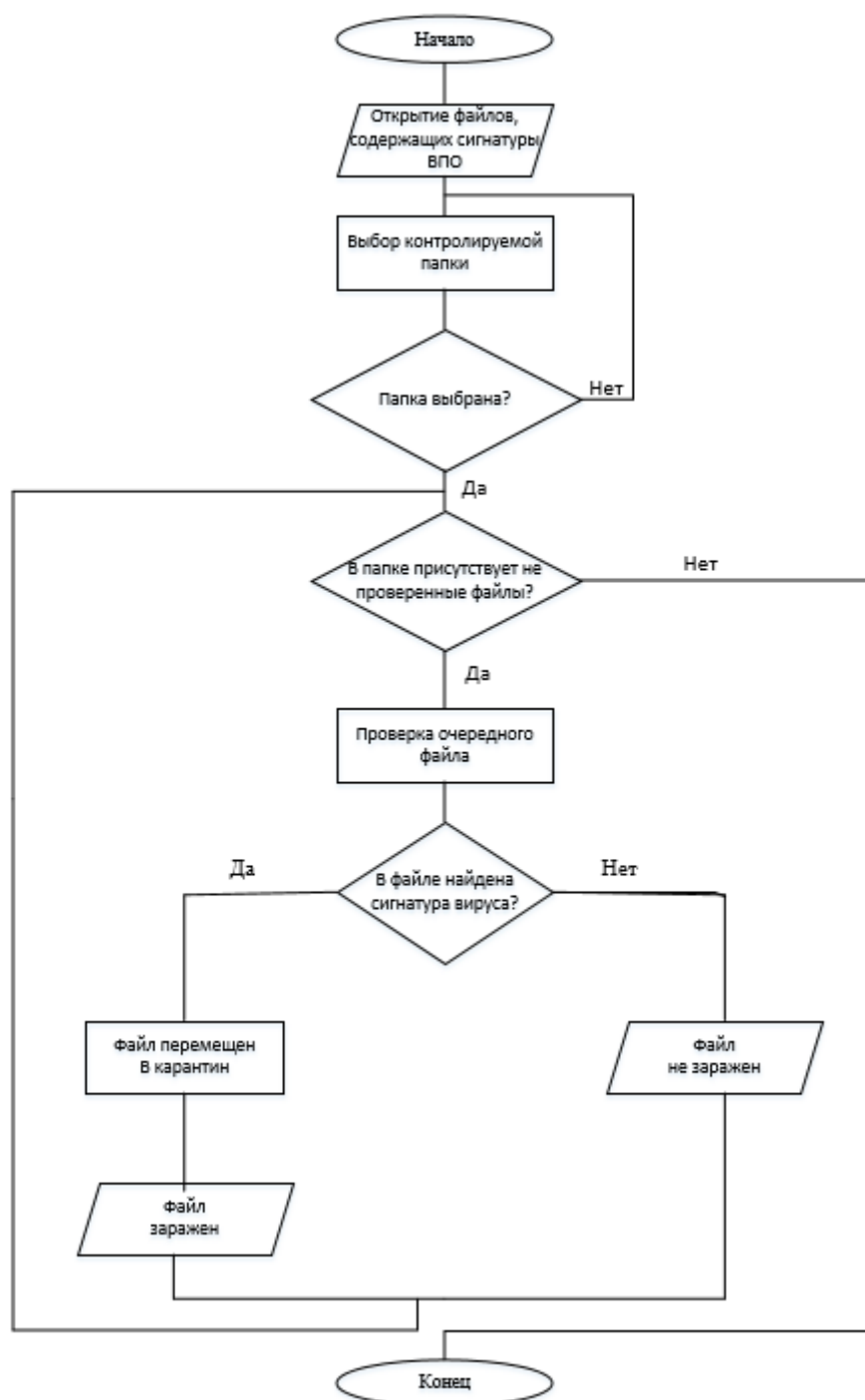


Рисунок 2.2 – Блок-схема алгоритма работы модуля выявления вредоносного кода.

2.3 Алгоритм работы модуля выявления изменений в файле.

После запуска ПО для выявления вредоносных программ в ОС Linux начинается отслеживание проводимых действий с файлами. Если файл был изменен, ПО для выявления вредоносных программ посылает сообщение с

вопросом пользователю: «Пользователь внес изменения в файле?». При отрицательном ответе файл помещается в папку «карантин», в котором доступ к файлам отсутствует, т.е. пользователь не может его использовать. При положительном ответе – документ остается в прежней папке. Соответствующие записи занесены в журнал событий. Блок-схема алгоритма модуля выявления изменений в файле представлена на рисунке 2.3.

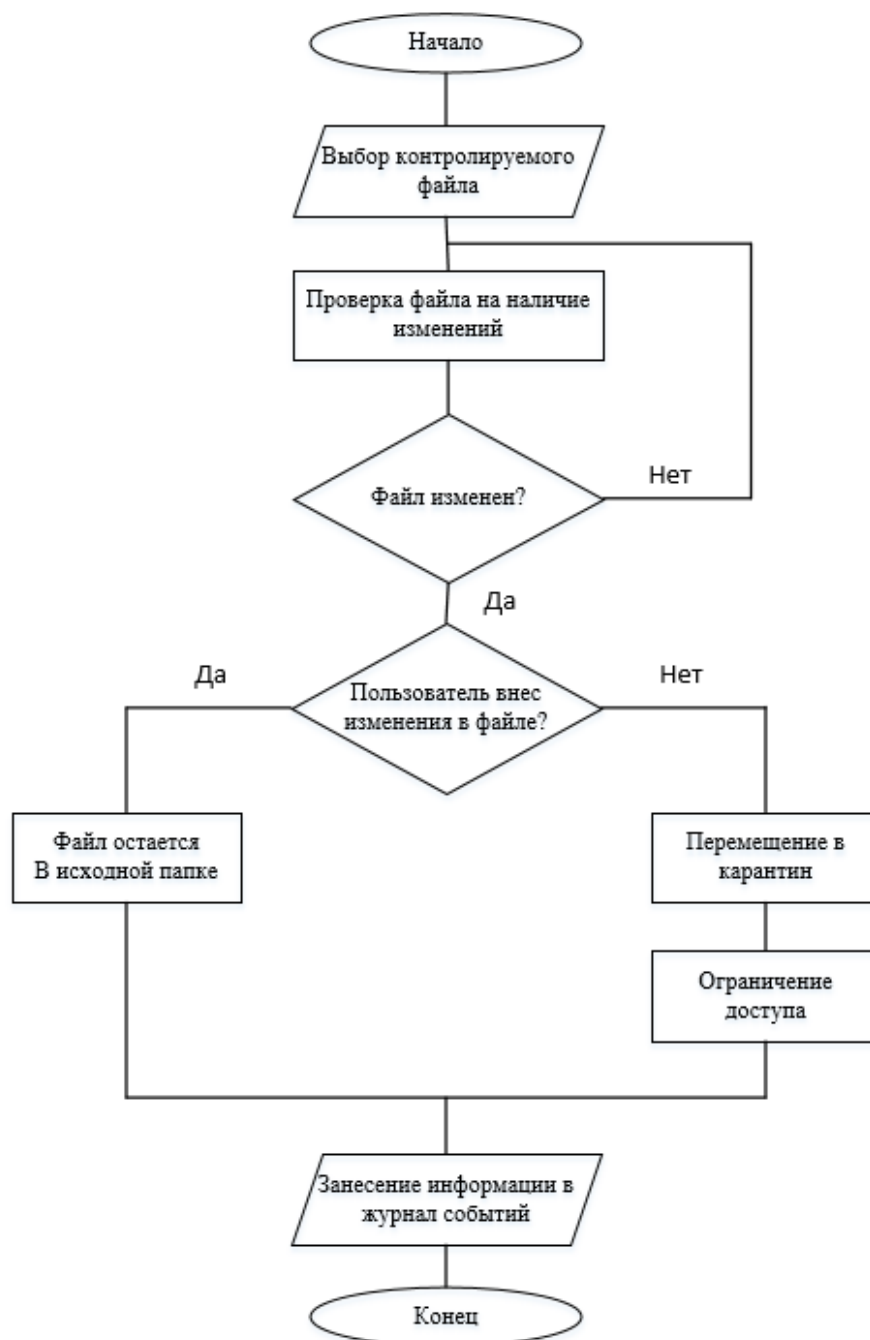


Рисунок 2.3 – Блок-схема алгоритма работы модуля выявления изменений в файле.

2.4 Алгоритм работы модуля карантин

Перенос в карантин осуществляется следующим образом. Вместо того, чтобы удалить файл сразу, он помещается в карантин для дополнительного анализа и проверки и сохранения его во избежание ошибочного удаления. После сканирования системы, ПО для выявления вредоносных программ может обнаружить файлы, которые представляют угрозу для ОС Linux. В этом случае файлы перемещаются в папку «карантин» и доступ к ним ограничивается. Т.е. такие файлы можно использовать только после их восстановления. При работе данного модуля при выполнении все действия записываются в журнал событий. Блок-схема алгоритма работы модуля перемещения в карантин представлена на рисунке 2.4.



Рисунок 2.4 – Блок-схема алгоритма работы модуля перемещение вредоносного файла в карантин

Если файл действительно является вредоносным пользователь может его удалить. В противном случае при ошибочном помещении в карантин он может быть восстановлен. Для извлечения файла из папки «карантин» пользователь должен выбрать нужный файл и ответить на вопрос в диалоговом окне: «Вы хотите восстановить этот файл?». При положительном ответе файл перемещается в папку своего исходного местонахождения. При отрицательном ответе он остается в карантине до принятия иного решения пользователем. Соответствующие записи заносятся в журнал событий. На рисунке 2.5 представлена блок-схема алгоритма работы модуля восстановления файла из карантина.

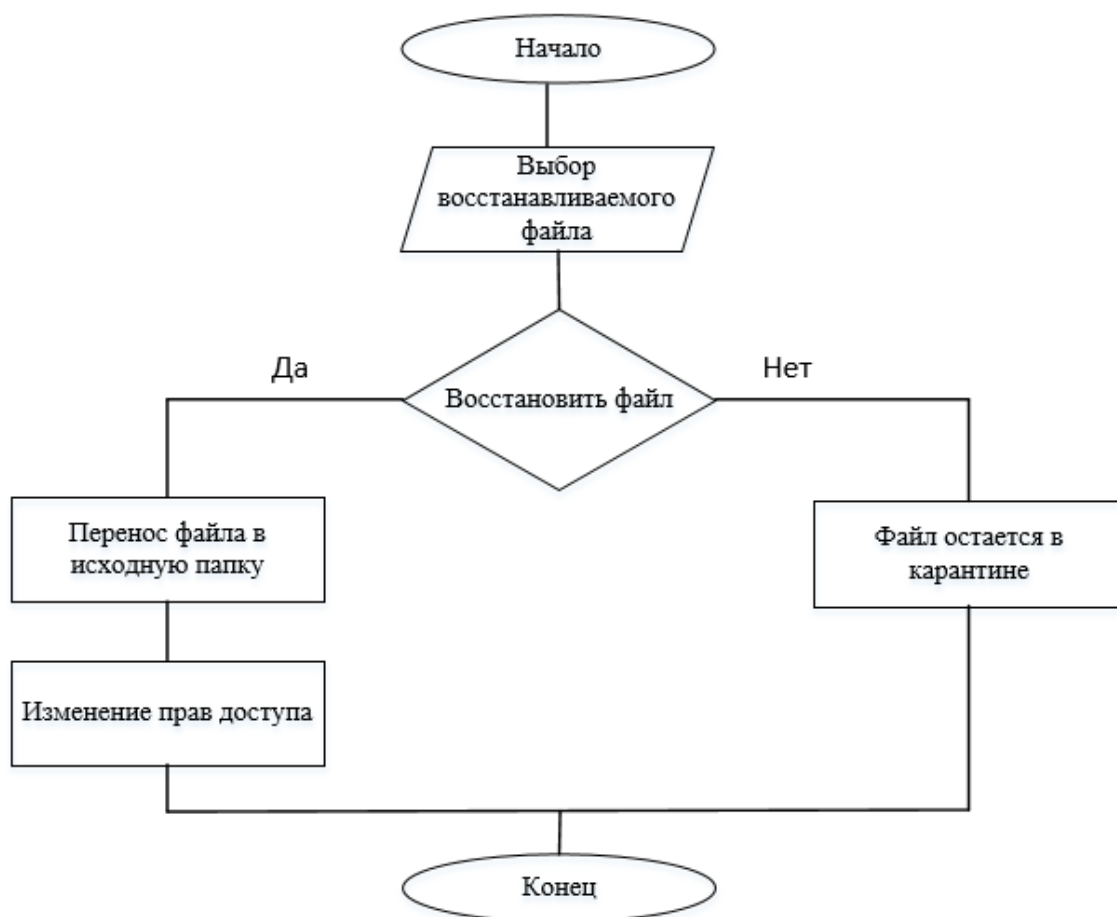


Рисунок 2.5 – Блок-схема алгоритма работы модуля восстановления файла из карантина.

2.5 Выводы по второй главе

Алгоритмы работы приложений являются важным средством для создания, разработки и отладки программного обеспечения. В данной главе были разработаны алгоритмы работы приложения для выявления вредоносного ПО в ОС Linux. Были разработаны следующие алгоритмы: общая работы ПО для выявления вредоносных программ, выявление вредоносного кода, выявление изменений в файле, модуль занесение и восстановление файла из карантина. Все разработанные алгоритмы модулей также представлены в виде блок-схем. В следующем разделе работы представлена программная реализация на их основе.

3. Программная реализация средства для выявления вредоносного ПО на основе разработанных алгоритмов в ОС Linux.

3.1 Выбор языка и среды разработки для выявления вредоносного ПО.

Язык программирования Python – это высокоуровневый, интерпретируемый, объектно-ориентированный язык программирования, который создан для быстрой программной реализации приложений. Он имеет простой и чистый синтаксис, который делает его легко читаемым и понятным. Python поддерживает многие парадигмы программирования, в том числе функциональное и процедурное программирование. Данный язык программирования имеет большое количество библиотек и фреймворков, которые обеспечивают широкие возможности для различных приложений, включая разработку программного обеспечения. Также Python имеет большое количество библиотек, которые обеспечивают широкие возможности для различных приложений, включая разработку программного обеспечения.

PyCharm - это кроссплатформенная интегрированная среда разработки (IDE) для языка программирования Python. PyCharm имеет широкий набор инструментов для разработки, отладки, тестирования, профилирования и оптимизации кода.

Функции PyCharm включают удобный редактор текста, поддержку многопоточности, поддержку Jupyter Notebook, создание документации кода, инструментарий для оптимизации производительности и многое другое.

3.2 Разработка интерфейса ПО для выявления вредоносных приложений.

На первом этапе необходимо разработать интерфейс приложения. Данный интерфейс должен быть интуитивно понятен, а также реализовывать все функциональные возможности. На листинге 1.1 представлена

программная реализация интерфейса. На рисунке 3.1 представлен разработанный интерфейс главного окна разработанного приложения.

Листинг 3.1. – Фрагмент программного кода модуля создания интерфейса.

```
class MainWindow(QMainWindow):

def __init__(self):
    super().__init__()
    self.setWindowTitle("Antivirus")
    log_label = QLabel("Журнал событий:")
    log_label.setFont(QFont("Arial", 14))
    self.log_list = QListView()
    self.log_list.setFixedSize(200, 500)
    self.log_model = QStandardItemModel()
    self.update_log_model()
    file_list_label = QLabel("Файлы в карантине:")
    file_list_label.setFont(QFont("Arial", 14))
    self.file_list = QListView()
    self.file_list.clicked.connect(self.show_restore_dialog)
    self.file_list.setFixedSize(300, 500)
    self.file_model = QStandardItemModel()
    self.update_file_model()
    input_label = QLabel("Путь:")
    input_label.setFont(QFont("Arial", 14))
    self.input_field = QLineEdit('/home/ok/Рабочий стол/test')
    self.start_button = QPushButton("Запустить")
    self.start_button.clicked.connect(self.start_process)
    self.stop_button = QPushButton("Остановить")
    self.stop_button.clicked.connect(self.stop_process)
    log_layout = QVBoxLayout()
    log_layout.addWidget(log_label)
    log_layout.addWidget(self.log_list)
    file_list_layout = QVBoxLayout()
    file_list_layout.addWidget(file_list_label)
    file_list_layout.addWidget(self.file_list)
    input_layout = QHBoxLayout()
    input_layout.addWidget(input_label)
    input_layout.addWidget(self.input_field)
    button_layout = QHBoxLayout()
    button_layout.addWidget(self.start_button)
    button_layout.addWidget(self.stop_button)
    right_layout = QVBoxLayout()
    right_layout.addLayout(input_layout)
    right_layout.addLayout(button_layout)
    main_layout = QHBoxLayout()
    main_layout.addLayout(log_layout)
    main_layout.addLayout(file_list_layout)
    main_layout.addLayout(right_layout)
    central_widget = QWidget()
    central_widget.setLayout(main_layout)
    self.setCentralWidget(central_widget)
    self.observer = Observer()
    self.event_handler = FileModifiedHandler()
```

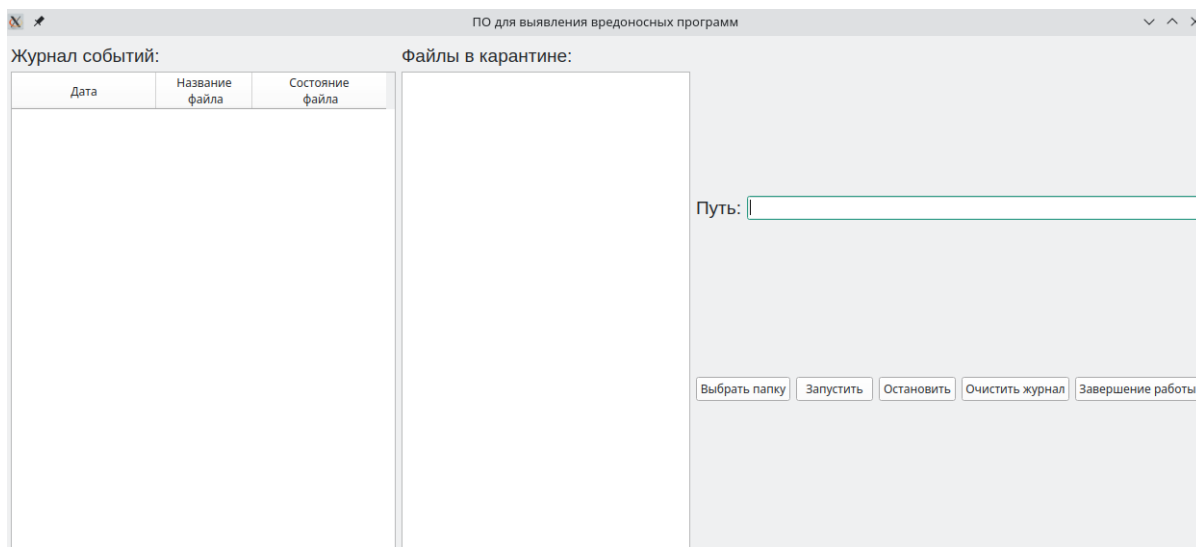


Рисунок 3.1 - Главный экран ПО для выявления вредоносных программ в ОС Linux.

3.3 Программная реализация модуля поиска сигнатур вредоносных приложений.

Поиск файлов, содержащих вредоносный код в ОС Linux происходит следующим образом. Сканер поочередно открывает файлы, с помощью функции `open()`. Далее читает информацию из них, с помощью команды `file.read()`. Если в проверяемых файлах находится сигнатура вредоносного ПО, то он перемещается в карантин. На листинге 3.2 представлен программный код данного модуля.

Листинг 3.2. – Программный код модуля поиска сигнатур вредоносных приложений в файлах.

```
def search_stop_words(file_path):
    for root, dirs, files in os.walk(file_path):
        for file_ in files:
            filep = root + "/" + file_
            with open(filep, 'r') as file:
                text = file.read()
                if any(word in text for word in stop_words):
                    move_file_to_quarantine(filep)
```

3.4 Программная реализация модуля «карантин».

После запуска приложения необходимо проверить наличие папки «карантин», и в случае ее отсутствия она создается с помощью команды `os.mkdir()`. Модуль `os.replace()` позволяет переместить зараженный или подозрительный файл в папку «Карантин». С помощью команды `os.chmod()`, осуществляется ограничение доступа к файлу. Это не позволит использовать данный файл пользователем, а также его выполнение, что позволит предотвратить дальнейшее заражение системы вредоносным ПО. На листинге 3.3. продемонстрирован программный код данной функции.

Листинг 3.3. – Программный код модуля перемещение вредоносного или подозрительного файла в карантин.

```
def move_file_to_quarantine(file_path):
    if not os.path.exists(quarantine_location):
        os.mkdir(quarantine_location)

    filename = get_file_name(file_path)
    new_file_path = os.path.join(quarantine_location, filename)
    print(new_file_path)
    os.replace(file_path, new_file_path)
    mode = 0
    os.chmod(new_file_path, mode)
```

Для восстановления файлов из карантина используется функция код, которой продемонстрирован на листинге 3.4. Аналогично с функцией перемещения файла в карантин работает и восстановление из карантина. Команда `os.replace()` и `os.chmod()` изменяет путь файла и восстанавливает права доступа соответственно.

Листинг 3.4. – Программный код модуля восстановления файла из карантина.

```
def restore_file(self, result):
    old_path, fname, = result[1], result[2]
    new_file_path = old_path
    file_path = quarantine_location + "/" + fname
    os.replace(file_path, new_file_path)
    mode = 777
    os.chmod(new_file_path, mode)
```

3.5 Программная реализация модуля выявления изменений в файле.

Для выявления изменений в файлах используется команда `event.srs_path()`. Она используется для определения, какой файл был изменен. Если изменения были сделаны не пользователем, файл помещается в карантин. Программный код данного модуля продемонстрирован на листинге 3.5.

Листинг 3.5. – Программный код модуля выявления изменений в файле.

```
def on_created(self, event):
    if event.is_directory:
        print(event)
    some_root = "/".join(event.src_path.split("/")[:-1])
    some_name = event.src_path.split("/")[-1]
    if some_name[0] == ".":
        some_name = some_name[1:]
    if len(some_name.split(".")) == 3:
        some_name = ".".join(some_name.split(".")[:2])
    new_path = some_root + "/" + some_name
    create_message_box(new_path)
def create_message_box(file_path):
    filename = get_file_name(file_path)
    msgBox = QMessageBox()
    msgBox.setText(f"Файл {filename} был изменён.")
    msgBox.setInformativeText("Это вы внесли изменения?")
    msgBox.setStandardButtons(QMessageBox.Yes|QMessageBox.No)
    buttonY = msgBox.button(QMessageBox.Yes)
    buttonY.setText('Да')
    buttonN = msgBox.button(QMessageBox.No)
    buttonN.setText('Нет')
    msgBox.setDefaultButton(QMessageBox.Yes)
    ret = msgBox.exec()

    if msgBox.clickedButton() == buttonY:
        msgBox.close()
    elif msgBox.clickedButton() == buttonN:
        move file to quarantine(file path)
```

3.6 Выводы по третьей главе

В данной главе представлена программная реализация ПО для выявления вредоносного ПО на основе разработанных алгоритмов для ОС Linux на языке программирования Python.

Правильная программная реализация является важным этапом разработки программного обеспечения, который позволяет создать

качественный и удобный в использовании программное обеспечение, отвечающий потребностям пользователей.

4. Обнаружение вредоносных программ в ОС Linux, с помощью разработанного программного обеспечения.

4.1 Выявление вредоносных приложений с помощью модуля поиска сигнатур

Интерфейс разработанного приложения выполнен в виде окна с соответствующими элементами управления. Для работы программы нужно выбрать папку, нажав на соответствующую кнопку. Именно эту папку будет сканировать приложение на предмет наличия вредоносных приложений. На рисунке 4.1 продемонстрировано окно выбора папки. Далее необходимо нажать на кнопку «Запустить» для начала работы сканера.

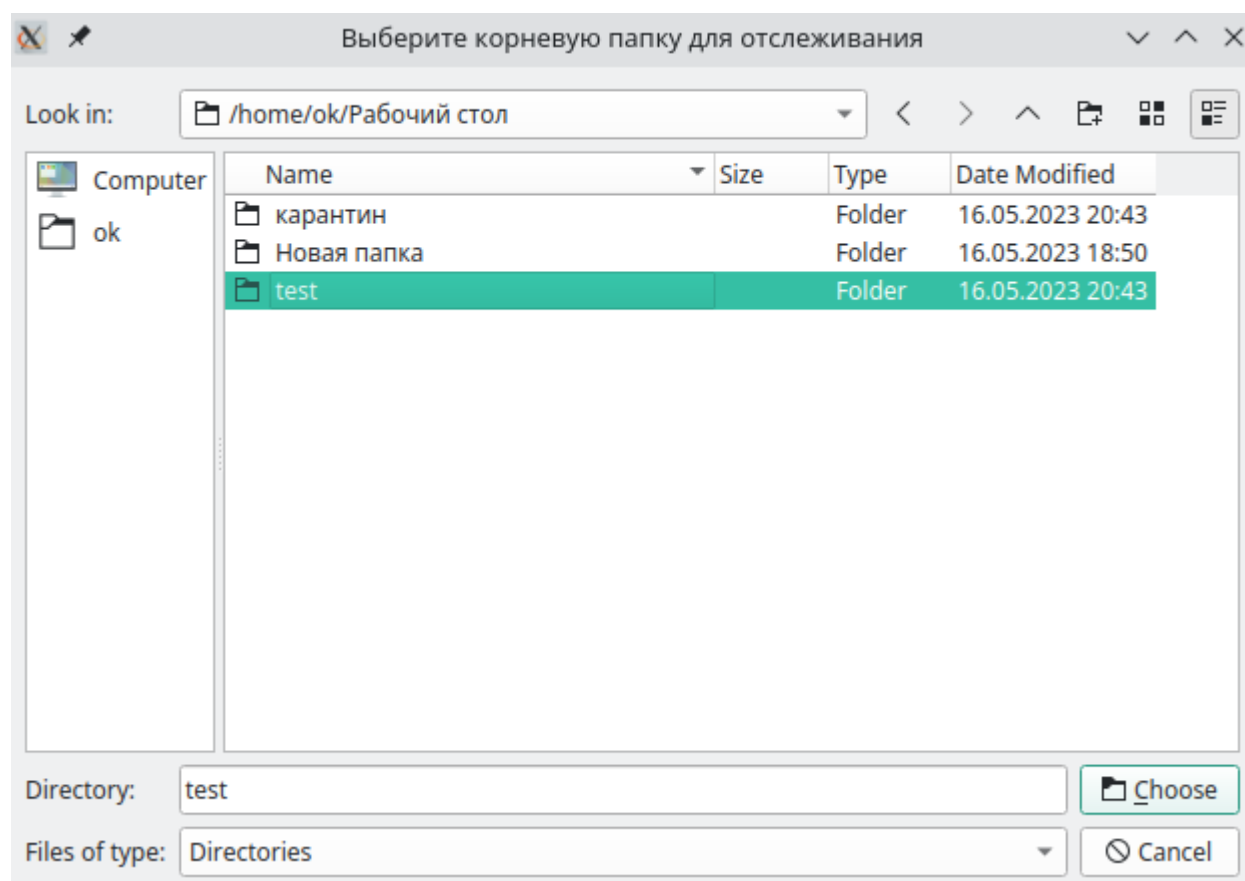


Рисунок 4.1 – Окно выбора папки.

На рисунке 4.2 представлена папка с файлами, с помощью которых моделируется наличие вредоносных файлов.

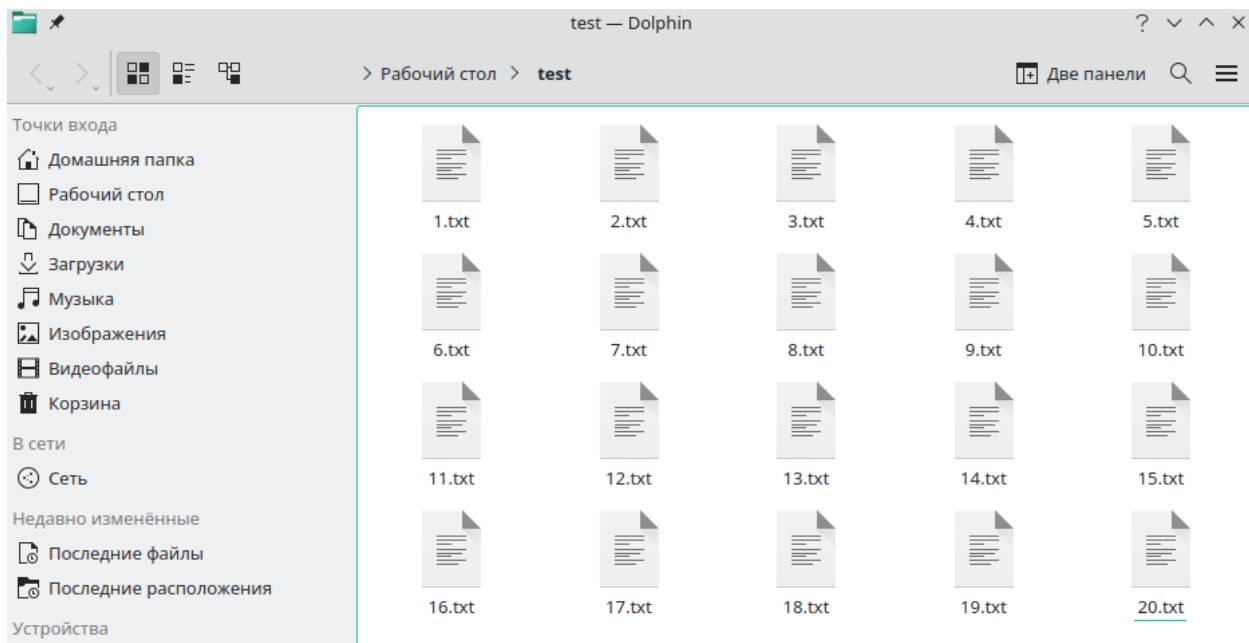


Рисунок 4.2 - Окно папки с файлами, с помощью которых моделируется наличие вредоносных файлов.

После запуска программы, для проведения тестирования на рабочем столе должна появиться папка «карантин». Но при работе ПО для выявления вредоносных программ данная папка должна находиться в более недоступном для пользователя месте. Это продемонстрировано на рисунке 4.3.

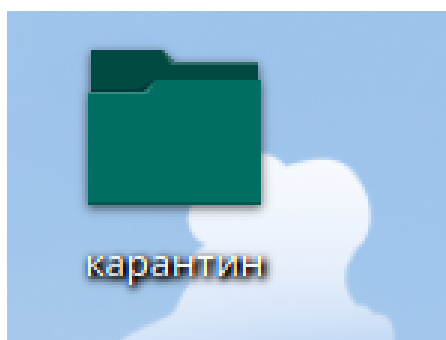


Рисунок 4.3 - Создание папки «карантин».

Для проверки работоспособности разработанного приложения принято, что зараженными файлами являются: «1», «3», «5», «7», «9», «11», «13», «15», «17», «19». В перечисленных файлах содержится последовательность которая

будет считаться сигнатурой вредоносной сигнатурой. При корректной работе приложения они должны быть перенесены в папку карантин и информация об этих событиях должна быть занесена в журнал событий. Это продемонстрировано на рисунке 4.4. Как видно, журнале событий заносятся соответствующие записи, включая записи о проверенных, но не зараженных файлах. В записях журнала событий присутствует данные о времени, когда файл был: добавлен в карантин, восстановлен из карантина, проверен ПО для выявления вредоносных программ.

Журнал событий:				Файлы в карантине:	
	Дата	Название файла	Состояние файла		
9	17/05/2023 00:39:07	18.txt	проверен	15.txt 7.txt 19.txt 1.txt 9.txt 3.txt 5.txt 17.txt 11.txt 13.txt	
10	17/05/2023 00:39:16	20.txt	проверен		
11	17/05/2023 00:47:14	2.txt	отправлен в карант		
12	17/05/2023 00:48:09	2.txt	восстановлен		
13	17/05/2023 00:48:11	2.txt	проверен		
14	17/05/2023 00:49:26	1.txt	отправлен в карант		
15	17/05/2023 00:50:00	3.txt	отправлен в карант		
16	17/05/2023 00:50:38	5.txt	отправлен в карант		
17	17/05/2023 00:51:12	7.txt	отправлен в карант		
18	17/05/2023 00:51:51	9.txt	отправлен в карант		
19	17/05/2023 00:52:25	11.txt	отправлен в карант		
20	17/05/2023 00:53:04	13.txt	отправлен в карант		
21	17/05/2023 00:53:39	15.txt	отправлен в карант		
22	17/05/2023 00:54:24	17.txt	отправлен в карант		
23	17/05/2023 00:54:55	19.txt	отправлен в карант		

Рисунок 4.4 - Запись в журнал информации о событиях.

На рисунке 4.5 продемонстрировано содержимое исходной папки в которой уже отсутствуют файлы, считающиеся зараженными файлов после проверки. Это означает, что в ней остались только незараженные файлы.

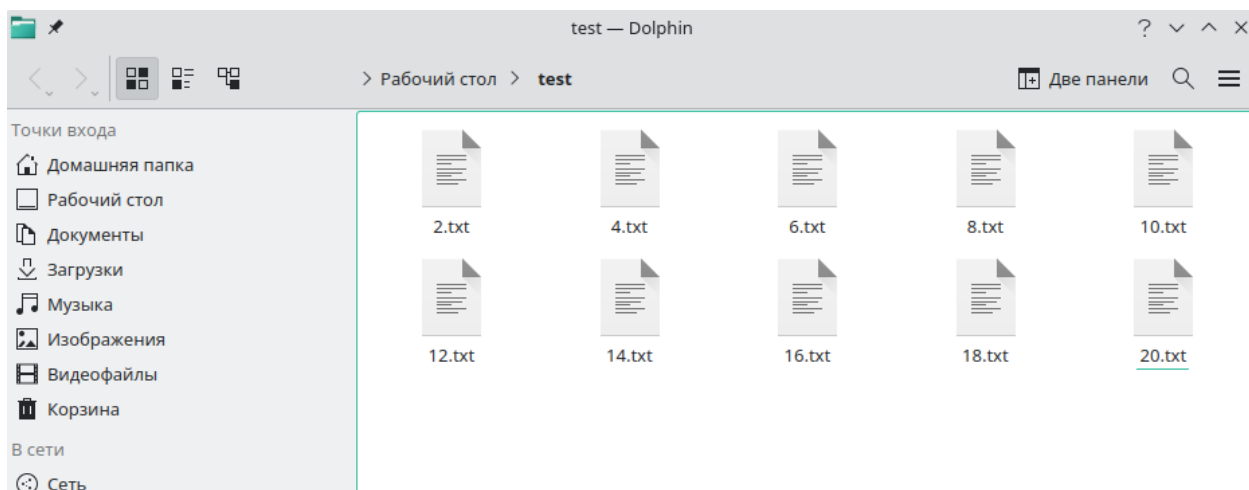


Рисунок 4.5 - Исходная папка после проверки.

На рисунке 4.6 можно увидеть содержимое папки «карантин». Все файлы, в которых была обнаружена последовательность, считавшаяся вредоносной были перенесены в данную папку верно.

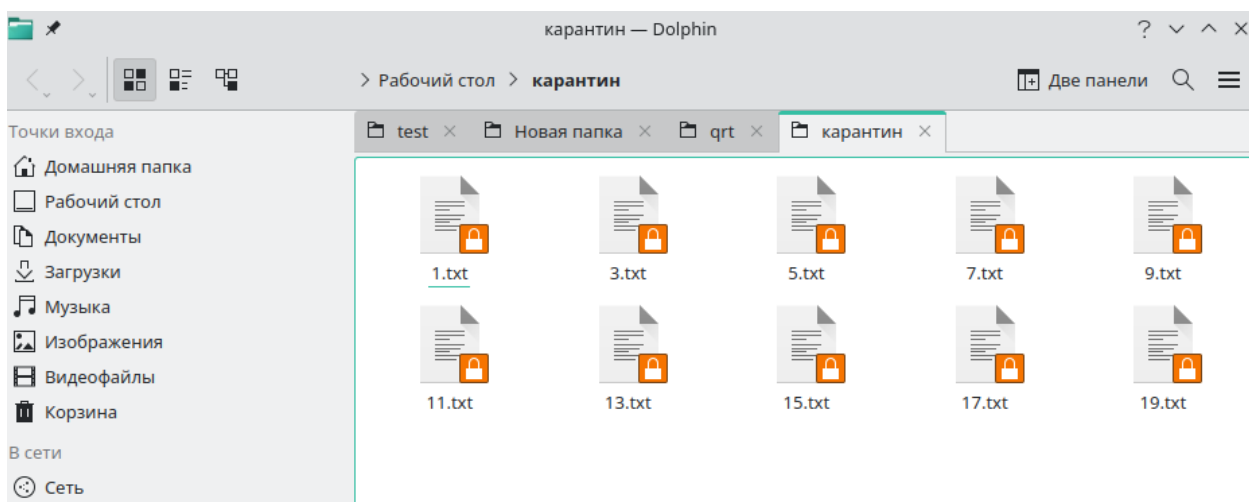


Рисунок 4.6 - Папка «карантин»

4.2 Обнаружение вредоносных приложений с помощью модуля выявления изменений в файле

Для проверки работоспособности программного обеспечения содержимое файла «2» изменяется. Разработанное ПО уведомляет об изменениях в файле и предоставляет выбор действий пользователю. Диалоговое окно, выводимое для пользователя, предоставляющее выбрать дальнейшее действие представлено на рисунке 4.7.

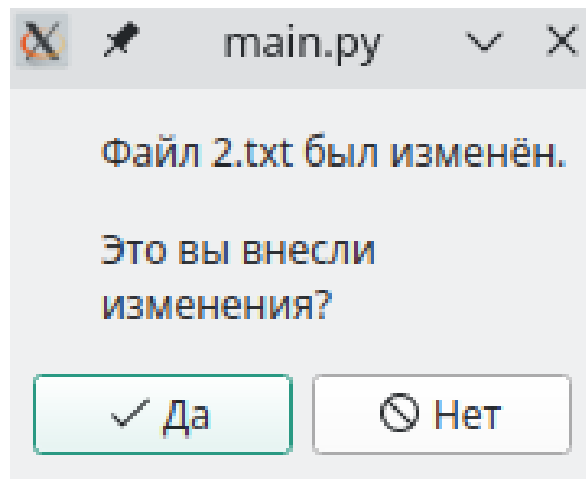


Рисунок 4.7 - Уведомление об изменении в файле.

Отрицательный ответ означает, что изменения в данном файле пользователь не осуществлял. И в таком случае данный файл перемещается в карантин. Соответствующая запись осуществляется в журнале событий. на рисунке 4.8 представлены результаты записи в журнал данного события. А также на рисунке 4.9 продемонстрировано содержимое папки «карантин». Представленные результаты говорят, что разработанное ПО работает корректно.

Журнал событий:			Файлы в карантине:
	Дата	Название файла	Состояние файла
1	17/05/2023 00:33:44	2.txt	проверен
2	17/05/2023 00:33:55	4.txt	проверен
3	17/05/2023 00:35:58	6.txt	проверен
4	17/05/2023 00:36:44	8.txt	проверен
5	17/05/2023 00:36:52	10.txt	проверен
6	17/05/2023 00:37:01	12.txt	проверен
7	17/05/2023 00:37:10	14.txt	проверен
8	17/05/2023 00:37:19	16.txt	проверен
9	17/05/2023 00:39:07	18.txt	проверен
10	17/05/2023 00:39:16	20.txt	проверен
11	17/05/2023 00:47:14	2.txt	отправлен в карантин
			2.txt

Рисунок 4.8 - Запись об изменении в файле.

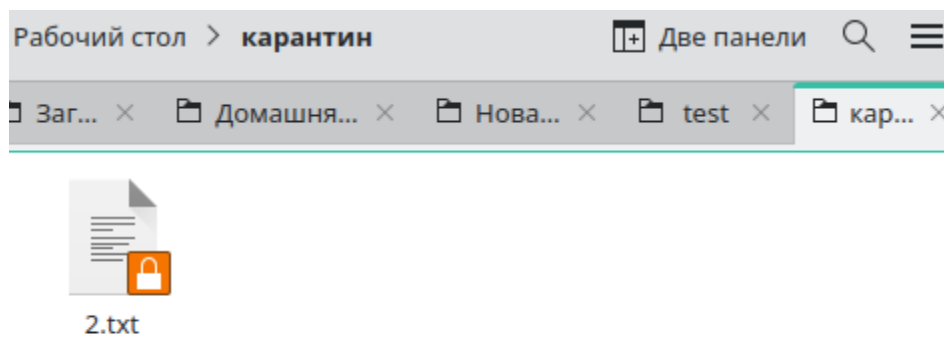


Рисунок 4.9 - Наличие зараженного файла.

Для того что бы выполнить восстановление файла из карантина пользователь выбирает нужный файл в журнале событий. Ему выводится диалоговое окно с сообщением для принятия решения касающегося восстановления файла. Это продемонстрировано на рисунке 4.10.

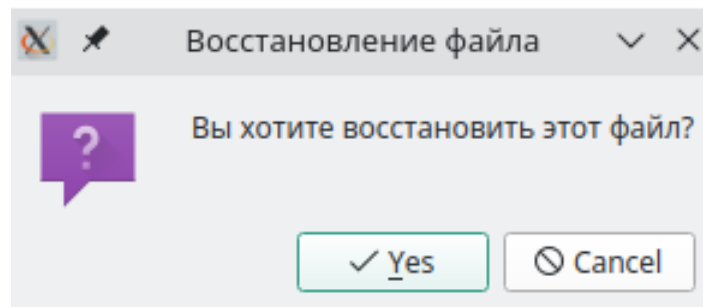


Рисунок 4.10 - Сообщение о восстановлении файла.

После восстановления файла в журнале событий записывается соответствующая запись. На рисунке 4.11 показана запись в журнале событий о восстановлении файла.

Журнал событий:			
	Дата	Название файла	Состояние файла
1	17/05/2023 00:33:44	2.txt	проверен
2	17/05/2023 00:33:55	4.txt	проверен
3	17/05/2023 00:35:58	6.txt	проверен
4	17/05/2023 00:36:44	8.txt	проверен
5	17/05/2023 00:36:52	10.txt	проверен
6	17/05/2023 00:37:01	12.txt	проверен
7	17/05/2023 00:37:10	14.txt	проверен
8	17/05/2023 00:37:19	16.txt	проверен
9	17/05/2023 00:39:07	18.txt	проверен
10	17/05/2023 00:39:16	20.txt	проверен
11	17/05/2023 00:47:14	2.txt	отправлен в карантин
12	17/05/2023 00:48:09	2.txt	восстановлен
13	17/05/2023 00:48:11	2.txt	проверен

Рисунок 4.11 - Запись о восстановлении файла.

На рисунке 4.12 продемонстрировано окно исходной папки в которой присутствует восстановленный файл.

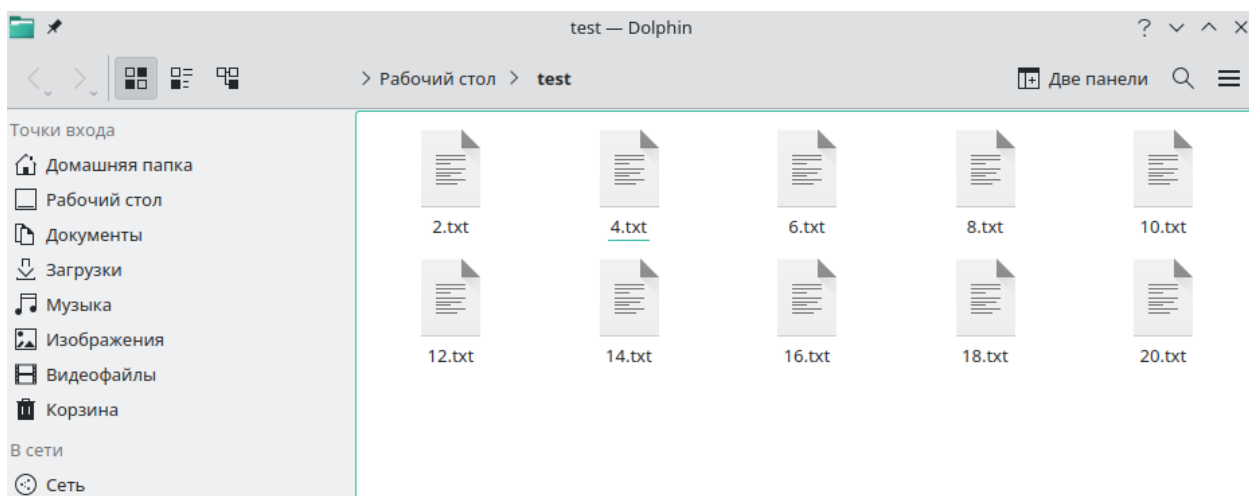


Рисунок 4.12 - Перенос файла в исходную папку из карантина.

Для защиты файлов, размещенных в карантине пользователям ограничивается доступ к данной папке. Это делается во избежание дальнейшего распространения вредоносного ПО. Пользователь ограничен в действиях и не может внести изменения в файл, который находится на карантине. рисунке 4.13 показано сообщение, выводимое пользователю при попытке открыть папку «карантин».

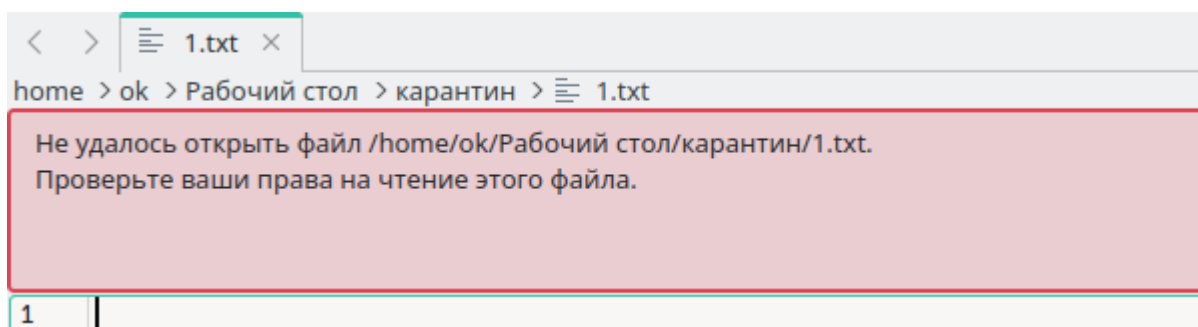


Рисунок 4.13 - Ограничение доступа.

В случае если в карантин файлы попадают ошибочно, то пользователь может восстановить все файлы из карантина, как показано выше. Если в

карантине не будет файлов, то в программе это будет отображено как показано на рисунке 4.14

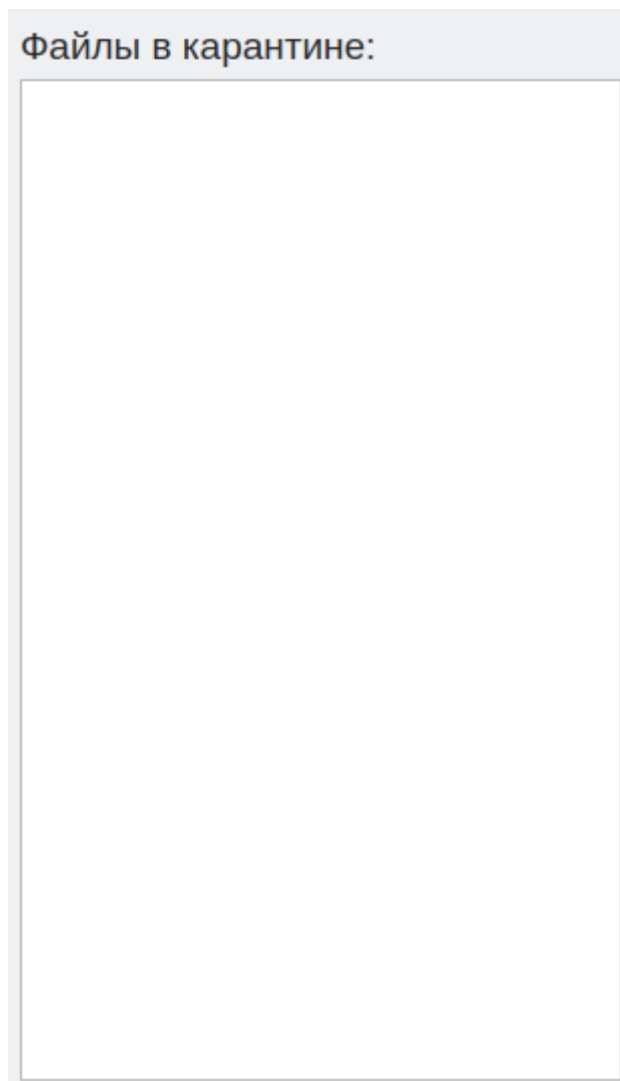


Рисунок 4.14 - Восстановление всех файлов.

На рисунке 4.15 продемонстрировано отсутствие в папке «карантин» файлов.

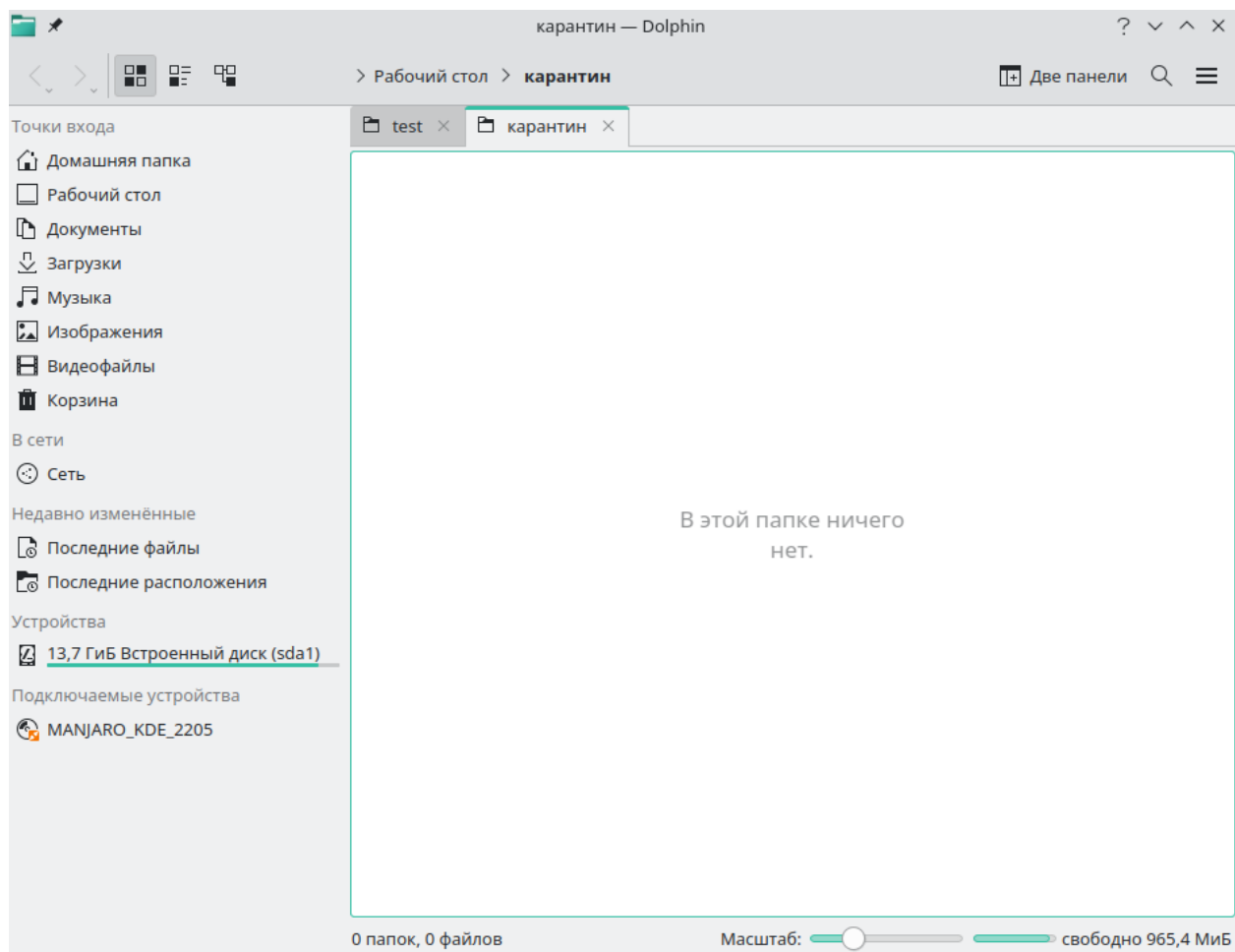


Рисунок 4.15 - Отсутствие файлов в папке «карантин».

Также в тестовом режиме программы пользователь может очистить журнал событий, нажав на соответствующую кнопку. Результат этого действия представлен на рисунке 4.16. Но в реальных условиях данная функция не должна использоваться, т.к. в таком случае может быть снижен уровень безопасности ПК.

Журнал событий:			
	Дата	Название файла	Состояние файла
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

Рисунок 4.16 - Результат очистки журнала событий.

Для завершения работы с приложением нужно нажать на кнопку «остановить» и закрыть программу.

4.3 Выводы по четвертой главе

В данном разделе было протестировано разработанное программное обеспечение для выявления вредоносных программ в ОС Linux. В ходе работы программы была проверена работа следующих модулей:

- Отслеживание изменений в файлах;
- Поиск сигнатуры вируса в файлах;
- Запись в журнал событий;
- Помещение файла в карантин;
- Восстановление файла из карантина;

- Очистка журнала событий.

С его помощью была выполнена проверка набора тестовых файлов.

Полученные результаты подтверждают корректность работы разработанного ПО. При моделировании заражения компьютера, было корректно выявлено тестовые файлы моделирующие зараженное ПО с помощью разработанного приложения использующего сигнатурный метод и метод обнаружения изменений в файлах. Для хранения тестовых файлов использовался карантин, к содержимому которого доступ пользователям и приложениям запрещен.

Заключение

Актуальность применения средств защиты от вредоносного ПО остается высокой в настоящее время, так как компьютерные вирусы и другие вредоносные программы продолжают представлять угрозу для компьютеров пользователей работающих на основе ОС Linux. Это связано с тем, что количество вредоносных программ постоянно растет, а их выявление и уничтожение становится с каждым днем более сложной задачей.

Цель работы - реализация программного обеспечения для выявления вредоносных программ в операционной системе Linux.

В первом разделе работы были рассмотрены методы борьбы с вредоносным ПО, особенности работы ПО для выявления вредоносных программ в ОС Linux. Во втором разделе представлены разработанные алгоритмы работы приложения для выявления вредоносного ПО в ОС Linux, а также их блок-схемы. В третьем разделе показаны результаты программной реализации приложения на основе разработанных алгоритмов на языке Python. В четвертом разделе было протестировано разработанное ПО для выявления вредоносных программ в ОС Linux. В ходе проверки выполнено моделирование, при котором разработанное ПО корректно обнаружило вредоносное ПО и поместило его в карантин.

По результатам выполненной работы можно сделать следующие выводы:

1. Сигнатурный метод не позволяет выявлять ВПО, если вредоносная программа изменяет свою сигнатуру или использует шифрование. Также могут происходить ложные срабатывания, когда не вредоносные файлы определяются как зараженные. Поэтому, более эффективное обнаружение вредоносных программ может быть достигнуто за счет комбинации различных методов, таких как сигнатурный, эвристический и поведенческий методы.

2. В случае отсутствия установленного ПО для выявления вредоносных программ компьютер остается не защищенным и уязвимым для атак вирусов и других вредоносных программ. Поэтому в настоящее время ПО для выявления вредоносных приложений является важным инструментом для

защиты компьютеров работающих на основе ОС Linux. В ходе работы были решены следующие задачи:

- Выполнен обзор существующих методов выявления вредоносного ПО;
- Разработаны алгоритмы работы приложения;
- Разработано ПО для выявления вредоносных программ в ОС Linux;
- Выполнена проверка работоспособности разработанного ПО.

Таким образом, все поставленные задачи полностью решены и цель работы достигнута.

Список используемых источников

1. Актуальные киберугрозы: 2022 года // Positive Technologies URL: <https://www.ptsecurity.com/> (дата обращения: 4.04.2023).
2. Вы атакованы — защищайтесь! // BYTE Россия URL: <https://bytemag.ru/> (дата обращения: 7.04.2023).
3. Н.Г. Булахов, В.Т. Калайда Методы обнаружения и обезвреживания саморазмножающихся вирусов // Доклады ТУСУРа, № 2 (18), часть 1, июнь 2008. - 2008. - №14. - С. 78-55.
4. доктор техн. наук, профессор А.В. Аграновский, кандидат техн. наук Р.А. Хади **НОВЫЙ ПОДХОД К ЗАЩИТЕ ИНФОРМАЦИИ – СИСТЕМЫ ОБНАРУЖЕНИЯ КОМПЬЮТЕРНЫХ УГРОЗ.** - 4-е изд. - г.Ростов-на-Дону: Просвещение, 2007. - 24 с.
5. Аркадьев С.А, Михайлюк И.С. Основы работы антивирусных программ // Антивирусная защита компьютерных систем. - 2020. - №7. - С. 44-52.
6. Костюкова Нина Ивановна Метод обнаружения вирусов // Альманах современной науки и образования. - 2021. - №23. - С. 72-86.
7. Защита от вредоносных программ и спама // YZTM.RU URL: <https://yztm.ru/> (дата обращения: 3.04.2023).
8. Т.И. Вишневская, О.К. Макаренко Виртуальная среда для контроля за действиями программ на операционной системе Linux // Информационные технологии в науке, образование и управление. - 2019. - №47. - С. 1-12.
9. Проскурин В.Г. Программно-аппаратные средства обеспечения информационной безопасности. Защита в операционных системах. –Москва: Радио и связь, 2000;
10. Афанасьева, Д.В. Сравнительный анализ антивирусного программного обеспечения / Д.Афанасьева // Известия Тульского государственного университета. Технические науки. 2021. №. 5. С. 216–218.
11. Атамкулова, М.Т., Саримсаков, А.А. Компьютерные вирусы и антивирусные программы / М.Атамкулова, А. Саримсаков // Известия Омского технологического университета. 2016. №4. С. 136–140.

12. Климентьев, К.Е. Компьютерные вирусы и антивирусы: взгляд программиста / К.Е. Климентьев. - 1 - е изд. - М.: ДМК Пресс, 2013. С. 658. ©
13. Афанасьева, Д.В. Компьютерные вирусы: специфика и противодействие // Наука, образование и культура. 2019. №. 3 (37). С. 11–12.
14. Ганижева, Н.Ж. Компьютерные вирусы и антивирусные программы // Молодой ученый. 2021. №. 33. С. 3–5
15. Котляров, В. П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2009. - 288 с
16. Левенталь, Л. Введение в микропроцессоры: Программное обеспечение, аппаратные средства, программирование / Л. Левенталь. - М.: Энергоатомиздат, 2019. - 464 с.
17. Мартин, Дж. Вычислительные сети и распределенная обработка данных: программное обеспечение, методы и архитектура / Дж. Мартин. - М.: Финансы и статистика, 2022. - 525 с.
18. Фленов, М.Е. Linux глазами хакера / М.Е. Фленов. - М.: БХВ-Петербург, **2023**. - 544 с.
19. Ф.Файтс, П.Джонстон, М.Кратц "Компьютерный вирус: проблемы и прогноз", Москва, "Мир", 2013 г.
20. Мостовой Д.Ю. "Современные технологии борьбы с вирусами" // Мир ПК. - №8. - 2014.
21. Гошко С.В «Энциклопедия по защите от вирусов» / Гошко С.В. –М.: СОЛОН-Пресс, 2012. – 304 с.
22. Алексеев В.М «Меры и методы обеспечения доверия к информационной безопасности»: уч.пособие/Алексеев В.М, Фатеев А.Г, Лупанов М.Ю. – Пенза: Изд-во Пенз.гос.ун-та, 2011. -244 с.
23. Фленов, М.Е. Linux глазами хакера / М.Е. Фленов. - М.: БХВ-Петербург, 2008. - 544 с.
24. Алексеев, П. П. Антивирусы. Настраиваем защиту компьютера от вирусов / П.П. Алексеев, Д.А. Козлов, Р.Г. Прокди. - М.: Наука и техника, 2008. - 399 с.

25. Алеексеев, П. Антивирусы. Настраиваем защиту компьютера от вирусов / П. Алеексеев, Д. Козлов, Р. Прокди. - М.: СПб: Наука и Техника, 2022. - 746 с.

Приложение

Листинг – Программный код для обнаружения вредоносных приложений.

```
import sys
import os
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
from PySide6 import QtCore
from PySide6.QtWidgets import QApplication, QMainWindow,
QMessageBox, QPushButton, QLabel, QWidget, QPlainTextEdit,
QListView, QHBoxLayout, QVBoxLayout, QLineEdit
import time
from PySide6.QtCore import Qt, QAbstractListModel
from PySide6.QtGui import QFont, QStandardItemModel,
QStandardItem
import sqlite3
quarantine_location = '/home/ok/Рабочий стол/qrt'
stop_words = ['rvhnjvnjbvjfvib123jsfbjkbvjkbjrb^%bxbvcg$%bnndxhcbhndbc']
stop_words = ['oddcmkcmiroogno,dclmbjnbknkmfkvmgfbjngj']
stop_words = ['trjgviotrjgmbviotjmbkmbjyjbnybnjynbjynbjynbjksmfldk;c,l;ds,cl;s
d,clrkrmrevmjrkvnjrn']
stop_words = ['jtigjirjijgrkmdkmclp;jfbgndmdkss,nchnckdkd6383jfkelsl*()fnrje
ngkjtwbghwlngwgnjgbjhewbhjgbur74892230239okrfgitjghjgh']
stop_words = ['htbvvh%^&*()nfvkf
vjhngvbi8493758467875935938nvhjnjekdcklcm']
stop_words = ['mkmkkmdcnjdc843hfdbhdfg&*&njrnjvnjrvn837ujcnn7*&*&']
stop_words = ['vnjfnvjnv849847857835hnjutdclmbjnbknkmfkvmgfbjngjvvnfvbui^*&
fjvnfjvn23ndjr(&*&']
stop_words = ['nnnbjbnjlgbnjltrn*&y66ybfhrvbhnf78e4ybvfhvhknk*&Hjnfvnjfnvjnv
jnoef94839857hfvbhf^*&(&*(hvbhbv9834jfncvua7y7uHU&unejkvnjerkv
nuigh479']
stop_words = ['fjnvbfjk 7846bhgdbdu^^*()*nhfv ejh3827392nfjv
rjnv']
stop_words = ['jvnjrvjinr754874578njhndhnhd8732782ffhfh&^&hfhjfhj^&^&bbbhv
bd$#&#nknkv76487329187ncjbghbvg*&HJNufb8238&']
conn = sqlite3.connect('my.db', check_same_thread=False)
cur = conn.cursor()

cur.execute(
    '''CREATE TABLE IF NOT EXISTS files
        (id INTEGER PRIMARY KEY,
         path VARCHAR,
         file_name VARCHAR,
         event VARCHAR)'''
)
conn.commit()
conn.close()
```

```

class FileModifiedHandler(FileSystemEventHandler):
    def on_created(self, event):
        if event.is_directory:
            print(event)
            some_root = "/".join(event.src_path.split("/")[:-1])
            some_name = event.src_path.split("/")[-1]
            if some_name[0] == ".":
                some_name = some_name[1:]
            if len(some_name.split(".")) == 3:
                some_name = ".".join(some_name.split(".")[:2])
            new_path = some_root + "/" + some_name
            create_message_box(new_path)
            time.sleep(5)

def create_message_box(file_path):
    filename = get_file_name(file_path)
    msgBox = QMessageBox()
    msgBox.setText(f"Файл {filename} был изменён.")
    msgBox.setInformativeText("Это          вы          внесли
изменения?")
    msgBox.setStandardButtons(QMessageBox.Yes |
QMessageBox.No)
    buttonY = msgBox.button(QMessageBox.Yes)
    buttonY.setText('Да')
    buttonN = msgBox.button(QMessageBox.No)
    buttonN.setText('Нет')
    msgBox.setDefaultButton(QMessageBox.Yes)
    ret = msgBox.exec()

    if msgBox.clickedButton() == buttonY:
        msgBox.close()
    elif msgBox.clickedButton() == buttonN:
        move_file_to_quarantine(file_path)

        conn          =          sqlite3.connect('my.db',
check_same_thread=False)
        cur = conn.cursor()
        cur.execute("INSERT INTO files (path, file_name,
event) VALUES(?, ?, ?)", (file_path, get_file_name(file_path),
'отправлен в карантин'))
        conn.commit()
        conn.close()
        global window
        window.update_log_model()
        window.update_file_model()
        msgBox.close()

def get_file_name(file_path):
    return file_path.split('/')[-1]

def move_file_to_quarantine(file_path):
    if not os.path.exists(quarantine_location):
        os.mkdir(quarantine_location)

```

```

        filename = get_file_name(file_path)
        new_file_path = os.path.join(quarantine_location, filename)
        print(new_file_path)
        os.replace(file_path, new_file_path)
        mode = 0
        os.chmod(new_file_path, mode)

def search_stop_words(file_path):
    for root,dirs,files in os.walk(file_path):
        for file_ in files:
            filep = root + "/" + file_
            with open(filep, 'r') as file:
                text = file.read()
                if any(word in text for word in stop_words):
                    move_file_to_quarantine(filep)
                    conn = sqlite3.connect('my.db',
check_same_thread=False)
                    cur = conn.cursor()
                    cur.execute("INSERT INTO files (path,
file_name, event) VALUES(?, ?, ?)",
                                (filep, get_file_name(filep),
'отправлен в карантин'))
                    conn.commit()
                    conn.close()
                    global window
                    window.update_log_model()
                    window.update_file_model()
                    QMessageBox.close()

def get_db_data():
    conn = sqlite3.connect('my.db', check_same_thread=False)
    cur = conn.cursor()
    cur.execute("SELECT * FROM files")
    rows = cur.fetchall()
    conn.close()
    return rows

class MainWindow(QMainWindow):

    def __init__(self):
        super().__init__()
        self.setWindowTitle("Antivirus")

        # Create widgets for left panel
        log_label = QLabel("Журнал событий:")
        log_label.setFont(QFont("Arial", 14))
        self.log_list = QListView()
        self.log_list.setFixedSize(200, 500)
        self.log_model = QStandardItemModel()
        self.update_log_model()

        # Create widgets for middle panel
        file_list_label = QLabel("Файлы в карантине:")

```

```

        file_list_label.setFont(QFont("Arial", 14))
        self.file_list = QListView()

        #on click show restore dialog

self.file_list.clicked.connect(self.show_restore_dialog)
        self.file_list.setFixedSize(300, 500)


        self.file_model = QStandardItemModel()
        self.update_file_model()


        # Create widgets for right panel
        input_label = QLabel("Путь:")
        input_label.setFont(QFont("Arial", 14))
        self.input_field = QLineEdit('/home/ок/Рабочий
стол/test')
        self.start_button = QPushButton("Запустить")
        self.start_button.clicked.connect(self.start_process)
        self.stop_button = QPushButton("Остановить")
        self.stop_button.clicked.connect(self.stop_process)


        # Create layout for left panel
        log_layout = QVBoxLayout()
        log_layout.addWidget(log_label)
        log_layout.addWidget(self.log_list)


        # Create layout for middle panel
        file_list_layout = QVBoxLayout()
        file_list_layout.addWidget(file_list_label)
        file_list_layout.addWidget(self.file_list)


        # Create layout for right panel
        input_layout = QHBoxLayout()
        input_layout.addWidget(input_label)
        input_layout.addWidget(self.input_field)


        button_layout = QHBoxLayout()
        button_layout.addWidget(self.start_button)
        button_layout.addWidget(self.stop_button)


        right_layout = QVBoxLayout()
        right_layout.addLayout(input_layout)
        right_layout.addLayout(button_layout)


        # Create main layout
        main_layout = QHBoxLayout()
        main_layout.addLayout(log_layout)
        main_layout.addLayout(file_list_layout)
        main_layout.addLayout(right_layout)


        # Create central widget and set layout
        central_widget = QWidget()
        central_widget.setLayout(main_layout)
        self.setCentralWidget(central_widget)

```



```

        self.observer = Observer()
        self.event_handler = FileModifiedHandler()

    def restore_file(self, result):
        old_path, fname, = result[1], result[2]
        new_file_path = old_path
        file_path = quarantine_location + "/" + fname
        os.replace(file_path, new_file_path)
        mode = 777
        os.chmod(new_file_path, mode)

        conn = sqlite3.connect('my.db', check_same_thread=False)
        cur = conn.cursor()
        cur.execute("INSERT INTO files (path, file_name, event)
VALUES(?, ?, ?)", (old_path, fname, 'восстановлен'))
        conn.commit()
        conn.close()

        self.update_file_model()
        self.update_log_model()
        print('файл восстановлен')

    def show_restore_dialog(self, index):
        result = QMessageBox.question(self, 'Восстановление
файла', 'Вы хотите восстановить этот файл?',

QMessageBox.StandardButton.Yes |
QMessageBox.StandardButton.Cancel)
        if result == QMessageBox.StandardButton.Yes:
            item = self.file_model.item(index.row())
            file_name_to_search = str(item.text())
            print(file_name_to_search)
            conn = sqlite3.connect('my.db',
check_same_thread=False)
            cur = conn.cursor()
            cur.execute("SELECT * FROM files WHERE file_name=?
ORDER BY id DESC LIMIT 1", (file_name_to_search,))
            result = cur.fetchone()
            if result:
                self.restore_file(result)
            else:
                print('Информации об этом файле нет в базе
данных')
        else:
            print('файл не восстановлен')

    def start_process(self):
        self.ROOT_DIR_TO_SCAN = self.input_field.text()
        self.observer.schedule(self.event_handler,
path=self.ROOT_DIR_TO_SCAN, recursive=True)
        self.observer.start()
        self.call_scanner()

```

```

def stop_process(self):
    if self.observer.is_alive():
        self.observer.stop()
    try:
        self.observer.join()
    except:
        pass
    print('сканер остановлен')
    self.observer = Observer()
    self.event_handler = FileModifiedHandler()

def call_scanner(self):
    search_stop_words(self.ROOT_DIR_TO_SCAN)
    print('запускаю сканер')

def get_file_names_for_manager(self):
    spisok = []
    if not os.path.exists(quarantine_location):
        os.mkdir(quarantine_location)
    for root,dirs,files in os.walk(quarantine_location):
        for file_ in files:
            spisok.append(file_)
    return spisok


def update_log_model(self):
    self.log_model.clear()
    for data in get_db_data():
        log_line = f"{data[2]} | {data[3]}"
        self.log_model.appendRow(QStandardItem(log_line))
        #self.file_model.appendRow(QStandardItem(data))
    self.log_list.setModel(self.log_model)

def update_file_model(self):
    self.file_model.clear()
    for file_name in self.get_file_names_for_manager():
        self.file_model.appendRow(QStandardItem(file_name))
    self.file_list.setModel(self.file_model)

if __name__ == "__main__":
    try:
        app = QApplication(sys.argv)
        window = MainWindow()
        window.show()
        # external_event = ExternalEvent()
        # external_event
        exit_code = app.exec()
    try:
        window.observer.stop()
        window.observer.join()
    except Exception as e:
        print(e)
    sys.exit(exit_code)
except Exception as e:
    print(e)
    print(e.args)

```



Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Ордена Трудового Красного Знамени
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Утверждаю
Зав. кафедрой

_____ 2023 г.

Раздаточный материал к бакалаврской работе
студента Никольской Д.И. гр. БПЗ1901 на тему

Разработка программного обеспечения для выявления вредоносных программ в операционной системе Linux

объем 11 страниц

Студент: Никольская Дарина Игоревна 
Научный руководитель: Симоныч А.Г. 

Москва 2023 г.

Министерство цифрового развития, связи и массовых
коммуникаций Российской Федерации Ордена Трудового Знамени
федеральное государственное бюджетное образовательное учреждение высшего
образования
«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ
И ИНФОРМАТИКИ»

**Разработка программного обеспечения
для выявления вредоносных программ в
ОС Linux**

Выпускная квалификационная работа бакалавра
Студента: Никольской Д.И.
Группа: БПЗ1901
Научный руководитель: доцент Симонян А.Г.

Цели и задачи работы

Цель работы:

Разработка программного обеспечения для выявления вредоносных программ в операционной системе Linux.

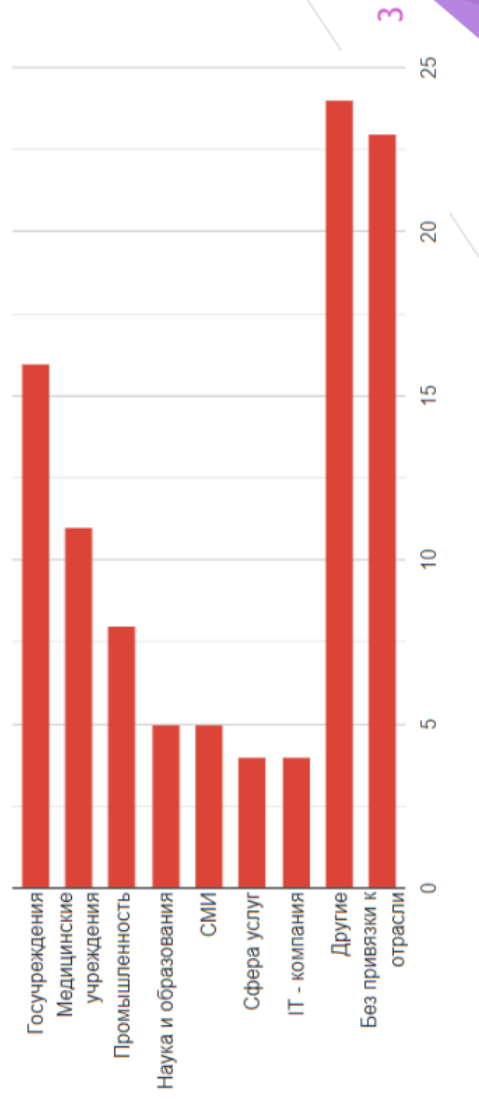
Для достижения данной цели необходимо решить следующие задачи:

- ▲ Выполнить обзор существующих методов выявления вредоносного ПО;
- ▲ Разработать алгоритмы работы приложения для выявления вредоносных программ в ОС Linux;
- ▲ Программно реализовать разработанные алгоритмы ПО для выявления вредоносных программ в ОС Linux;
- ▲ Проверить работоспособность разработанного ПО.

Актуальность

По мере появления огромного числа вредоносного ПО, обнаружение вторжений и реагирование на них имеет решающее значение для специалистов. Анализ вредоносных программ стал обязательным навыком по борьбе с вредоносным ПО и целевыми атаками.

Статистика атак, с использованием вредоносных программ в 2022 году, показывает, что они являются серьезной проблемой не только для физических лиц, но и для организаций. На рисунке представлен график категорий жертв, подвергшихся атакам в 2022 году.



Методы выявления вредоносного ПО

Определить наличие вредоносного ПО на компьютере можно по различным признакам. К ним можно отнести следующие:

- ❑ Замедление работы компьютера;
- ❑ Уменьшение объема оперативной памяти;
- ❑ Зависание, перезагрузка или блокировка компьютера;
- ❑ Ошибки при работе ОС или прикладных программ;
- ❑ Изменение размеров файлов и др.

Методы проанализированные в работе:

- ❑ Сигнатурный метод;
- ❑ Эвристический метод;
- ❑ Поведенческий метод.

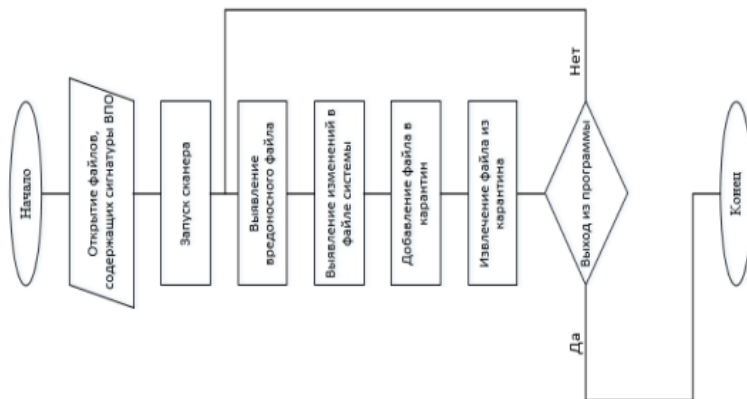
Принципы работы приложения для выявления вредоносного ПО

Монитор – отслеживает все манипуляции с файлами в режиме реального времени. Находит и обезвреживает вредоносное ПО до того, как оно успеет инфицировать систему.

Проверка в режиме реального времени обеспечивает непрерывность работы ПО для выявления вредоносных программ. Это реализуется с помощью обязательной проверки всех действий, совершаемых другими программами и самим пользователем, на предмет вредоносности вне зависимости от их исходного расположения – будь то свой жесткий диск, внешние носители информации, сетевые ресурсы или оперативная память.

Во многих приложениях, осуществляющих выявление вредоносных программ среди вспомогательных средств имеется технология – карантин. Карантин - это место для хранения файлов, считающихся подозрительными или вредоносными. Эта область позволяет избежать потери данных в случае некорректных действий ПО для выявления вредоносных программ по отношению к файлам.

Блок-схема алгоритма работы приложения для выявления вредоносного ПО в ОС Linux



Интерфейс разработанного ПО

ПО для выявления вредоносных программ

Журнал событий:

Дата	Название файла	Состояние файла
------	----------------	-----------------

файлы в карантине:

Путь:

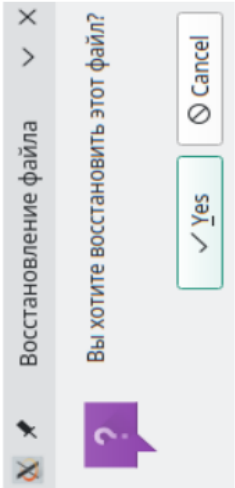
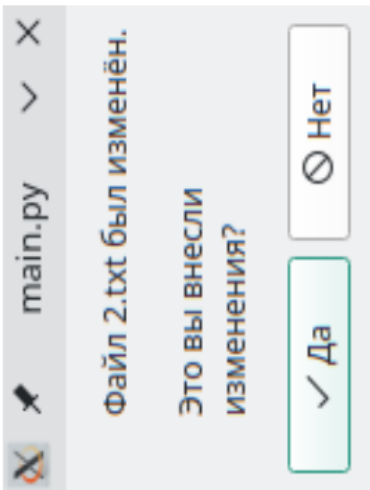
Выборать папку Запустить Остановить Очистить журнал Завершение работы

Результаты работы разработанного приложения для выявления вредоносного ПО в ОС Linux

Тестирование модуля поиска сигнатур ВПО в файлах.

Журнал событий:			Файлы в карантине:	
Дата	Название файла	Состояние файла		
9 17/05/2023 00:39:07	18.txt	проверен	15.txt	
10 17/05/2023 00:39:16	20.txt	проверен	7.txt	
11 17/05/2023 00:47:14	2.txt	отправлен в карант	19.txt	
12 17/05/2023 00:48:09	2.txt	восстановлен	1.txt	
13 17/05/2023 00:48:11	2.txt	проверен	9.txt	
14 17/05/2023 00:49:26	1.txt	отправлен в карант	3.txt	
15 17/05/2023 00:50:00	3.txt	отправлен в карант	5.txt	
16 17/05/2023 00:50:38	5.txt	отправлен в карант	17.txt	
17 17/05/2023 00:51:12	7.txt	отправлен в карант	11.txt	
18 17/05/2023 00:51:51	9.txt	отправлен в карант	13.txt	
19 17/05/2023 00:52:25	11.txt	отправлен в карант		
20 17/05/2023 00:53:04	13.txt	отправлен в карант		
21 17/05/2023 00:53:39	15.txt	отправлен в карант		
22 17/05/2023 00:54:24	17.txt	отправлен в карант		
23 17/05/2023 00:54:55	19.txt	отправлен в карант		

Тестирование модуля выявления изменений в файлах



Журнал событий:

	Дата	Название файла	Состояние файла
1	17/05/2023 00:33:44	2.txt	проверен
2	17/05/2023 00:33:55	4.txt	проверен
3	17/05/2023 00:35:58	6.txt	проверен
4	17/05/2023 00:36:44	8.txt	проверен
5	17/05/2023 00:36:52	10.txt	проверен
6	17/05/2023 00:37:01	12.txt	проверен
7	17/05/2023 00:37:10	14.txt	проверен
8	17/05/2023 00:37:19	16.txt	проверен
9	17/05/2023 00:39:07	18.txt	проверен
10	17/05/2023 00:39:16	20.txt	проверен
11	17/05/2023 00:47:14	2.txt	отправлен в карантин
12	17/05/2023 00:48:09	2.txt	восстановлен
13	17/05/2023 00:48:11	2.txt	проверен

Заключение

В случае отсутствия установленного ПО для выявления вредоносных программ компьютер остается не защищенным и уязвимым для атак вирусов и других вредоносных программ. Поэтому в настоящее время ПО для выявления вредоносных приложений является важным инструментом для защиты компьютеров работающих на основе ОС Linux.

В работе все поставленные задачи полностью решены и цель достигнута.

Спасибо за внимание!

11