Главное управление по образованию

Минского областного исполнительного комитета

Учреждение образования

«Новопольский государственный аграрно-экономический колледж»

Специальность 2-40 01 01 «Программное обеспечение информационных технологий»

Специальность 2-40 01 01 35 «Программное обеспечение обработки экономической и деловой информации»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К ДИПЛОМНОМУ ПРОЕКТУ НА ТЕМУ**

**Программное средство защиты от файловых вирусов**

Разработал \_\_\_\_\_\_\_\_\_\_\_ Н.И. Володько

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_ В.Л. Захарич

Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_ Т.Ю. Платонова

Консультанты \_\_\_\_\_\_\_\_\_\_\_ Т.Е. Волотовская

\_\_\_\_\_\_\_\_\_\_\_ Т.Л. Лобзанюк

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_ И.В. Усенко

Дипломный проект защищен с отметкой \_\_\_\_\_\_\_\_\_\_\_

Председатель ГКК

Новое Поле, 2020

Главное управление по образованию

Минского областного исполнительного комитета

Учреждение образования

«Новопольский государственный аграрно-экономический колледж»

Специальность *2-40 01 01 «Программное обеспечение информационных технологий»*

группа *2218*

УТВЕРЖДАЮ

Заместитель директора

по учебной работе

Т.Н. Крумкач

*«21» декабря 2020 г.*

**ЗАДАНИЕ**

**на дипломный проект**

учащегося *Володько Никиты Ивановича*

(фамилия, имя, отчество)

1. Тема проекта *Программное средство защиты от файловых вирусов*

Закреплена приказом по учреждению образования *21.02.2020 №123*

2. Срок сдачи учащимся законченного проекта *27.02.2021*

3. Исходные данные по проекту информация о вирусах

4. СОСТАВ ПРОЕКТА

а) Пояснительная записка

Введение

Основная часть

1 Назначение и область применения

1.1 Характеристика предметной области

1.2 Назначение программного средства

1.3 Описание аналогов

2 Технические характеристики

2.1 Постановка задачи на разработку программного средства

2.2 Описание программного средства

2.2.1 Общие сведения

2.2.2 Функциональное назначение

2.2.3 Описание логической структуры

2.2.4 Используемые технические средства

2.2.5 Вызов и загрузка

2.2.6 Входные данные

2.2.7 Выходные данные

2.2.8 Рекомендации по использованию

3 Ожидаемые технико-экономические характеристики

3.1 Экономический раздел

3.2 Охрана труда

Заключение

Список используемых источников

б) Графическая часть проекта

Лист 1 Диаграмма состояния

Лист 2 Диаграмма вариантов использования

Лист 3 Диаграмма деятельности

5. Консультанты (с указанием разделов, относящихся к ним)

|  |  |
| --- | --- |
| Раздел | Консультант |
| Теоретическая часть | Захарич В.Л. |
| Моделирование и программирование | Захарич В.Л. |
| Экономический раздел | Лобзанюк Т.Л. |
| Охрана труда | Волотовская Т.Е. |

6. Календарный график работы на весь период проектирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов дипломного проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Выбор среды программирования | 18-24.12.2020 |  |
| 2 | Постановка задачи | 18-21.12.2020 |  |
| 3 | Введение | 21-24.12.2020 |  |
| 4 | Проектирование | 04-12.01.2021 |  |
| 5 | Охрана труда | 04.01-06.02.2021 |  |
| 6 | Экономический раздел | 29.01-06.02.2021 |  |
| 7 | Разработка проекта | 04.01-10.02.2021 |  |
| 8 | Описание проекта | 08-15.02.2021 |  |
| 9 | Тестирование | 15-19.02.2021 |  |
| 10 | Заключение. Графическая часть | 18-22.02.2021 |  |
| 11 | Оформление пояснительной записки | 22-27.02.2021 |  |

7. Дата выдачи задания: 28 декабря 2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Захарич

(подпись)

Консультанты: Т.Е. Волотовская

(подпись)

Т.Л. Лобзанюк

(подпись)

Задание принял к исполнению Н.И. Володько

(подпись)

Председатель цикловой комиссии Т.Ю. Платонова

(подпись)



**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ДП2218.02.102.081ПЗ

Разраб.

Володько Н.

Провер.

Захарич В.

Реценз.

Н. Контр.

Усенко И.

Утверд.

Программное средство защиты от файловых вирусов

Лит.

Листов

50

НГАЭК, 2020

[ВВЕДЕНИЕ 7](#_Toc63623596)

[ОСНОВНАЯ ЧАСТЬ 9](#_Toc63623597)

[1. Назначение и область применения 9](#_Toc63623598)

[1.1 Описание предметной области 9](#_Toc63623599)

[1.2 Назначение программного средства 21](#_Toc63623600)

ПРИЛОЖЕНИЯ



# **ВВЕДЕНИЕ**

Все возрастающие усилия по созданию технологий защиты данных их уязвимость в современных условиях не только не уменьшается, но и постоянно возрастает. Актуальность проблем, связанных с защитой информации все более усиливается. Это требует от пользователя персонального компьютера знаний о природе вирусов, способах заражения вирусами и защиты от них. Проблема защиты информации является многоплановой и комплексной и охватывает ряд важных задач. Например, конфиденциальность данных, которая обеспечивается применением различных методов и средств. Интенсивное развитие современных информационных технологий, и в особенности сетевых технологий, создает для этого все предпосылки.

Вирусы, получившие широкое распространение в компьютерной технике, взбудоражили весь мир. Все чаще в средствах массовой информации появляются сообщения о различного рода проделках компьютерных злоумышленников, о появлении все более совершенных, саморазмножающихся программ. Несмотря на принятые во многих странах законы о борьбе с компьютерными преступлениями и разработку специальных программных средств защиты от вирусов, количество новых программных вирусов постоянно растет.

Объект исследования – программное средство.

Предмет исследования – программное средство защиты от файловых вирусов.

Цель дипломного проекта – разработать программное средство защиты операционной системы от файловых вирусов.

Задачи на дипломный проект:

* изучить методы защиты операционной системы от файловых вирусов;
* разработать структуру программного средства;
* разработать и протестировать программное средство;
* рассчитать себестоимость и отпускную цену программного средства;
* изучить тему по охране труда «».

Пояснительная записка к дипломному проекту состоит из 8 таблиц, 27 рисунков, 4 диаграмм и 15 источников.

В введении указана актуальность, объект, предмет исследования, цель и задачи дипломного проекта.

В введении раскрыта актуальность темы, определен объект исследования, предмет исследования, цель и задачи дипломного проекта.

В разделе «Назначение и область применения» раскрыта актуальность темы дипломного проекта, описаны виды вирусов, методы и способы борьбы с ними, проведен анализ аналогов.

В разделе «Проектирование» описана структура программного средства, выбор аппаратных средств, основные алгоритмы, методы разработки программных модулей, реализация взаимосвязи компонентов программного средства, а также рекомендации по использованию и описание графического интерфейса.

В экономическом разделе произведен расчет экономических характеристик.

В разделе «Охрана труда» изучена тема «Организация общественного контроля за состоянием охраны труда», в которой раскрыты такие вопросы как организация общественного контроля условий безопасности труда и общественный контроль за состоянием охраны труда на объекте при проектируемой деятельности.

В заключении подведены итоги разработки дипломного проекта, выделены преимущества, указаны перспективы модернизации и сопровождения разработанного программного средства.

# **ОСНОВНАЯ ЧАСТЬ**

1. Назначение и область применения

1.1 Описание предметной области

15 февраля 1946 года, была представлена новейшая и прогрессивная ЭВМ Eniac. Спустя 5 лет после её представления, американец Джон фон Нейман, заложил основы теории самовоспроизводящихся механизмов, он предложил метод создания таких механизмов, эти основы были опубликованы. Спустя некоторое время, учёный Ф. Ж. Шталь, используя материалы статьи Джона фон Неймана, запрограммировал на машинном языке биокибернетическую модель, в которой существа двигались, питаясь ненулевыми словами. Так же, в 1961 году, в фирме Bell Labs изобрели игру, в которой несколько ассемблерных программ, называемых организмами, загружались в память компьютера. Организмы, созданные одним игроком (то есть принадлежащие к одному виду), должны были уничтожать представителей другого вида и захватывать жизненное пространство. Победителем считался тот игрок, чьи организмы захватывали всю память или набирали наибольшее количество очков.

В сентябре 1984 года, был предложен новый термин «вирус». А уже зимой того же года, были опубликованы первые антивирусные утилиты. Они позволяли проанализировать текст загрузочного модуля и выявлять все текстовые сообщения и «подозрительные» участки кода (команды прямой записи на диск и др.). Благодаря своей простоте (фактически использовался только контекстный поиск) и эффективности они получили значительную популярность. Так же эти утилиты перехватывали операции записи и форматирования, выполняемые через [BIOS](https://ru.wikipedia.org/wiki/BIOS). Спустя некоторое время, в начале 1985 года, был создан первый резидентный антивирус, перехватывающий попытки записи на [дискеты](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D0%B5%D1%82%D0%B0) и [винчестер](https://ru.wikipedia.org/wiki/%D0%96%D1%91%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA), и осуществляющий блокировку всех операций (запись, [форматирование](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), выполняемые через BIOS.

Очередным этапом развития вирусов считается [1987 год](https://ru.wikipedia.org/wiki/1987_%D0%B3%D0%BE%D0%B4). К этому моменту получили широкое распространение сравнительно дешёвые компьютеры IBM PC, что привело к резкому увеличению масштаба заражения компьютерными вирусами. Именно в 1987 вспыхнули сразу три крупные эпидемии компьютерных вирусов. Первая эпидемия была вызвана вирусом Brain. За один год, этот вирус смог поразить более 18 тысяч компьютеров по всему миру. Вирус перезагружал компьютеры, на которых использовалось не лицензионное программное обеспечение. Он так же являлся первым невидимым вирусом, при попытке чтения заражённого сектора, он подставлял его незаражённый оригинал.

Вторая вирусная эпидемия была связанна с развитием глобальной сети интернет. В 1988 году, Робертом Моррисом был создан первый сетевой червь. Занимал червь всего 60 килобайт. Проникая в компьютер жертвы, червь подключал компоненты, позволяющие раскрывать пароли, имеющиеся в системе, что в свою очередь, позволяло программе маскироваться под задачу легальных пользователей системы, на самом деле занимаясь размножением и рассылкой копий. Он поразил свыше 6200 компьютеров. В результате вирусной атаки большинство сетей вышло из строя на срок до пяти суток. Компьютеры, выполнявшие коммутационные функции, работавшие в качестве файл-серверов или выполнявшие другие функции обеспечения работы сети, также вышли из строя.

Третья вирусная эпидемия была наиболее крупная. Появился вирус Datacrime, спящий вирус. Вирус распространялся и никак не выдавал себя, но начиная с 12 октября производил разрушение файловой системы на заражённых компьютерах. Всего было заражено более 100 тысяч ПЭВМ.

Начиная с 2010 года, проблема вирусов начинает принимать глобальный масштаб. 17 июня 2010 года, был обнаружен ранее не известный вирус Stuxnet. Вся проблема заключалась в том, что он был обнаружен не только на персональных компьютерах пользователей, но и в промышленных системах, управляющих автоматизированными производственными процессами. Это первый в истории вирус, который перехватывал и модифицировал информационный поток между программируемыми логическими контроллерами и рабочими станциями. Уникальность заключалась в том, что вирус мог физически разрушать инфраструктуру. Поэтому он мог быть использован как средство несанкционированного сбора данных и диверсий. Размер его, составлял всего 500 КБ.

Существует огромное количество различных вирусов. В таблице 1.1.1 приведена общая классификация типов вирусов.

Таблица 1.1.1 ­­– Общая классификация типов вирусов

|  |  |
| --- | --- |
| Наименование | Описание |
| Анти-антивирусный вирус | Компьютерная вирусная программа, объектом нападения которой являются антивирусные программы |
| Вариант вируса, штамм, модификация | Модифицированный вариант одного и того же вируса |
| Вирусная программа-червь | Паразитическая программа, обладающая механизмом саморазмножения. Программа способна размножать свои копии, но не поражать другие компьютерные программы. Проникает на компьютер из сети и рассылает свои функциональные копии на другие компьютерные сети |
| Вирусный мистификатор | Не являющееся вирусом почтовое сообщение. На компьютер пользователя мистификация приходит в виде письма, написанного в подчеркнуто нейтральном тоне, в котором указывается на якобы распространяющийся новый вирус. Пользователю предлагается найти некий файл с помощью поискового средства Windows и удалить его с диска |
| Вирусы-спутники | Формально являются файловыми вирусами. Не внедряются в исполняемые программы. Такие вирусы используют особенность системы, позволяющую программному файлу с тем же названием, но другим расширением действовать с разными приоритетами. Такие вирусы могут быть резидентными и маскировать файлы-двойники |
| Дроппер | Файл-носитель, устанавливающий вирус в систему. Техника, иногда используемая вирусописателями для сокрытия вирусов от антивирусных программ |
| Зоологический вирус | Вирусы, которые существуют только в антивирусных лабораториях, в коллекциях исследователей вирусов |
| Полиморфные вирусы | Вирусы с самомодифицирующимися расшифровщиками, использующие помимо шифрования кода специальную процедуру расшифровки, изменяющую саму себя в каждом новом экземпляре вируса, что ведет к отсутствию у него байтовых сигнатур |
| MtE вирусы | Полиморфные вирусы, созданные с помощью генератора полиморфизма |
| Резидентный вирус | Постоянно присутствующий в памяти вирус, написанный, как правило на низкоуровневом языке. Такие вирусы обладают возможностью эффективно заражать программы и противодействовать антивирусным средствам. |
| Скрипт-вирус | Вирус, написанный на языках Visual Basic, Basic Script, Java Script. На компьютер пользователя чаще всего проникают в виде почтовых сообщений, содержащих во вложениях файлы-сценарии. Программы на языках Visual Basic и Java Script могут располагаться как в отдельных файлах, так и встраиваться с HTML документ и в таком случае интерпретироваться с браузером, причем не только с удаленного сервера, но и с локального диска |
| Стелс-вирус | Вирусные программы, предпринимающие специальные действия для маскировки своей деятельности с целью сокрытия своего присутствия в зараженных объектах. Делятся на 2 типа: руткит и буткит |
| Шифрованные вирусы | Вирусы, которые сами шифруют свой код для затруднения их дизассемблирования и обнаружения в файле, памяти или секторе. Каждый экземпляр такого вируса будет содержать только короткий общий фрагмент – процедуру расшифровки – который можно выбрать в качестве сигнатуры. В случае каждого инфицирования он автоматически зашифровывает себя, и каждый раз по-разному. Таким способом вирус пытается избежать обнаружения вирусными программами. |

В таблице 1.1.2 приведена классификация именований вирусов

Таблица 1.1.2 – Классификация наименований вирусов

|  |  |
| --- | --- |
| Наименование | Описание |
| High Level Language вирусы | В данную группу объединяют вирусы, написанные на языках программирования высокого уровня, таких как C#, Java, Pascal |
| Троянские программы | В данную группу объединяют вирусы, которые осуществляют несанкционированные пользователем действия на его компьютере. |
| Silly-вирусы | Вирусы, которые не обладают никакими особенными характеристиками, вследствие чего нет необходимости присвоить таким вирусам особенные названия |
| Макровирусы | В данную группу объединяют вирусы, использующие особенности файлов офисных программ, а так же встроенные макроязыки данных приложений. |
| Скрипт-вирусы | В данную группу объединяют вирусы, написанные на различных интерпретируемых языках. Разделяют на 9 классов: VBS, JS, Wscript, HTML, Perl, PHP, IRC, Java, Bat. |
| Узкоспециализированное ПО | В данную группу объединяют вирусы, разработанные для инфицирования файлов конкретных программ. Выделяют 3 класса: ACAD, AutoLisp, SWF |
| Суффиксы | В эту группу входит 4 класса: вирусный конструктор, инсталлятор вируса. |
| Для DoS-атак | В данную группу объединяют вредоносные программы, разработанные для осуществления DoS атак с целью довести вычислительную технику до отказа в обслуживании |
| Потенциально опасное ПО | В данную группу объединяют программы, неспособные к самовоспроизведению и представляющие потенциальную опасность. В группу входит 4 классов: программы-шутки, хакерские утилиты, рекламные вирусы, программы, которые могут быть использованы злоумышленниками |
| Вирусы для разных ОС | В данную группу объединяют вредоносные программы, разработанные для определения ОС. Выделяют 19 классов. |
| Опасное ПО | Программы данной группы характеризируют как опасные и разделяют на 3 класса: Backdoor, Trojan, Exploit |
| Инструменты создания вредоносного ПО | В данную группу объединяют программы, которые используются для генерации или создания вирусов (вирусные конструкторы). |

В таблице 1.1.3 приведена классификация вирусов по видам заражаемых объектов.

Таблица 1.1.3 – Классификация вирусов по видами заражаемых объектов.

|  |  |
| --- | --- |
| Наименование | Описание |
| Файловые вирусы | Вирусы, заражающие двоичные файлы (в основном исполняемые файлы и динамические библиотеки). Чаще всего, такие файлы имеют расширение EXE, COM, DLL, SYS. Так же могут инфицировать файлы с расширениями DRV, BIN, OVL, OVY. Такие вирусы внедряются в файлы операционной системы, активируются при запуске пораженной программы и затем распространяются |
| Загрузочные (бутовые) вирусы | Вирусы, которые заражают загрузочные записи дискет, разделов жестких дисков, а также MBR жестких дисков |
| Макрокомандные вирусы (макровирусы) | Вирусы, заражающие файлы документов, используемые приложениями Microsoft Office и другими программами, допускающие наличие макрокоманд. Благоприятным фактором распространения вируса служит то, что все основные компоненты Microsoft Office могут содержать встроенные программы на полнофункциональном языке программирования, а в Microsoft Word эти макросы автоматически запускаются при открытии любого документа, его закрытии, сохранении и т.д. Учитывая то, что копирование макросов из документа в документ (в частности в общий шаблон) выполняется всего одной командой |

В зависимости от среды обитания вирусы можно разделить на:

– сетевые;

– файловые;

– загрузочные;

– файлово-загрузочные.

Сетевые вирусы распространяются по различным компьютерным сетям.

Файловые вирусы внедряются главным образом в исполняемые модули, а именно в файлы, имеющие расширения com и exe. Загрузочные вирусы внедряются в загрузочный сектор диска (Boot-сектор) или в сектор, содержащий программу загрузки системного диска (Master Boot Record). Файлово-загрузочные вирусы заражают как файлы пользователя, так и загрузочные сектора подключенных жестких дисков.

По способу заражения вирусы делятся на:

– резидентные;

– нерезидентные.

Резидентный вирус при заражении (инфицировании) компьютера оставляет в оперативной памяти свою резидентную часть, которая потом перехватывает обращение операционной системы к объектам заражения (файлам, загрузочным секторам дисков) и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения или перезагрузки компьютера.

Нерезидентные вирусы не заражают память компьютера и являются активными ограниченное время.

По степени воздействия вирусы можно разделить на следующие виды:

– неопасные, не мешающие работе компьютера, но уменьшающие объем свободной оперативной памяти и памяти на дисках, действия таких вирусов проявляются в каких-либо графических или звуковых эффектах;

– опасные вирусы, которые могут привести к различным нарушениям в работе компьютера;

– очень опасные, воздействие которых может привести к потере программ, уничтожению данных, стиранию информации в системных областях диска.

Заражению часто подлежат файлы программ из пакета Microsoft office. В эти файлы при заражении всегда вставляются макросы, которые создаются на языке Visual Basic, который поддерживается в подавляющем большинстве продуктов Microsoft office.

По особенностям алгоритма вирусы трудно классифицировать из-за большого разнообразия. Простейшие вирусы – паразитические, они изменяют содержимое файлов и секторов диска и могут быть достаточно легко обнаружены и уничтожены. Вирусы-репликаторы, называемые червями, которые распространяются по компьютерным сетям, вычисляют адреса сетевых компьютеров и записывают по этим адресам свои копии. Вирусы-невидимки, называемые стелс-вирусами, которые очень трудно обнаружить и обезвредить, так как они перехватывают обращения операционной системы к пораженным файлам и секторам дисков и подставляют вместо своего тела незараженные участки диска. Наиболее трудно обнаружить вирусы-мутанты (полиморфные вирусы), содержащие алгоритмы шифровки-расшифровки, благодаря которым копии одного и того же вируса не имеют ни одной повторяющейся цепочки байтов. Имеются и так называемые квазивирусные или «троянские» программы, которые хотя и не способны к самораспространению, но очень опасны, так как, маскируясь под полезную программу, могут разрушить загрузочный сектор и файловую систему дисков.

Так же следует не забывать про атаки с использованием внешних аппаратных устройств. Примером такой атаки является атака BadUSB, основанная на уязвимости USB устройств. Она устроена следующим образом: для определённого контроллера съёмного накопителя создаётся специальная прошивка, которая содержит код вируса, и уязвимый контроллер (съёмный накопитель) прошивается этой прошивкой. Модифицированная прошивка может исполнять любые команды злоумышленника, при этом оставаясь незамеченным в системе, так как все действия идут от эмулированного устройства ввода-вывода, например, сетевой карты, клавиатуры, мыши, или к примеру загрузочным устройством. Атака действует на все устройства, в которых есть USB порт, вне зависимости от установленной операционной системы. Единственное условие – должен быть установлен драйвер поддержки USB устройств.

Вирусы способны заражать такие объекты, как:

– исполняемые файлы;

– загрузчик системы;

– файлы документов.

Вирусы, которые заражают файлы называются файловыми. Вирус в зараженных исполняемых файлах начинает свою работу при запуске той программы, в которой он находится.

Вирусы, которые заражают загрузчик операционной системы и главную загрузочную запись называются загрузочными. Обычно загрузочные вирусы разделены на 2 части: первая часть находится в главной загрузочной записи, вторая находится в другом участке диска, или в кластере в области данных диска. Разделение на 2 части обусловлено тем, что в главную загрузочную запись невозможно записать что-либо большое.

С ростом угрозы, появлялось всё больше программ, решений и методов, для борьбы с ними, самыми актуальными методами обнаружения угроз являются:

– сигнатурный анализ;

– эвристический анализ;

– метод эмуляции исполнения программного кода.

При проверке компьютера на предмет угроз, в первую очередь применяется сигнатурный анализ. Он выполняется путем проверки содержимого анализируемого объекта на предмет наличия в нем сигнатур уже известных вирусов. Сигнатурой называется непрерывная конечная последовательность байт, необходимая и достаточная для однозначной идентификации угрозы. При этом сравнение содержимого исследуемого объекта с сигнатурами производится не напрямую, а по их контрольным суммам, что позволяет значительно снизить размер записей в вирусных базах, сохранив при этом однозначность соответствия и, следовательно, корректность обнаружения угроз и лечения инфицированных объектов. Записи в вирусных базах составляются таким образом, что благодаря одной и той же записи можно обнаруживать целые классы или семейства угроз.

Работа эвристического анализатора основывается на наборе эвристик (предположений, статистическая значимость которых подтверждена опытным путем) о характерных признаках вредоносного и, наоборот, безопасного исполняемого кода. Каждый признак кода имеет определенный вес (т. е. число, показывающее важность и достоверность этого признака). Вес может быть как положительным, если признак указывает на наличие вредоносного поведения кода, так и отрицательным, если признак не свойственен компьютерным угрозам. На основании суммарного веса, характеризующего содержимое объекта, эвристический анализатор вычисляет вероятность содержания в нем неизвестного вредоносного объекта. Если эта вероятность превышает некоторое пороговое значение, то выдается заключение о том, что анализируемый объект является вредоносным.

Эвристический анализатор также использует технологию Fly-Code – универсальный алгоритм распаковки файлов. Этот механизм позволяет строить эвристические предположения о наличии вредоносных объектов в объектах, сжатых программами упаковки (упаковщиками), причем не только известными разработчикам продукта, но и новыми, ранее не исследованными программами. При проверке упакованных объектов также используется технология анализа их структурной энтропии, которая позволяет обнаруживать угрозы по особенностям расположения участков их кода. Эта технология позволяет на основе одной записи вирусной базы произвести обнаружение набора различных угроз, упакованных одинаковым полиморфным упаковщиком.

Поскольку эвристический анализатор является системой проверки гипотез в условиях неопределенности, то он может допускать ошибки как первого (пропуск неизвестных угроз), так и второго рода (признание безопасной программы вредоносной). Поэтому объектам, отмеченным эвристическим анализатором как «вредоносные», присваивается статус «подозрительные».

Метод эмуляции исполнения программного кода используется для обнаружения полиморфных и шифрованных вирусов, когда использование поиска по контрольным суммам сигнатур неприменимо или значительно усложнено из-за невозможности построения надежных сигнатур. Метод состоит в имитации исполнения анализируемого кода при помощи эмулятора – программной модели процессора и среды исполнения программ. Эмулятор оперирует с защищенной областью памяти (буфером эмуляции). При этом инструкции не передаются на центральный процессор для реального исполнения. Если код, обрабатываемый эмулятором, инфицирован, то результатом его эмуляции станет восстановление исходного вредоносного кода, доступного для сигнатурного анализа.

Сканирование является наиболее традиционным методом поиска вирусов. Оно заключается в поиске уже ранее обнаруженных вирусов. Антивирусные программы-сканеры, способные удалить обнаруженные вирусы, обычно называют полифагами. Недостаток метода сканирования в том, что невозможно обнаружить полиморфные вирусы, полностью меняющие свой код. Для этого нужно использовать более сложные алгоритмы поиска, включающие эвристический анализ. Сканнеры не способны защитить компьютер от новых вирусов, поэтому они не эффективны.

Наиболее эффективным методом обнаружения вирусов является эвристический анализ. Эвристический анализ зачастую используется совместно с сигнатурным анализом для поиска шифрующихся и полиморфных вирусов. В большинстве случаев эвристический анализ позволяет также обнаруживать и ранее неизвестные вирусы. В этом случае, скорее всего их лечение будет невозможно.

Самым ресурсо-затратным методом, является метод обнаружения изменений файлов. Заражая компьютер, вирус делает изменения на жестком диске, на обнаружении таких изменений и основывается этот метод. В системе может существовать одновременно очень большое количество процессов, которые так или иначе вносят изменения в файловую систему компьютера. Антивирусной программе приходится проверять и фильтровать каждую такую операцию ввода-вывода.

По статистике компании Dr. Web только 30% обнаруженных угроз приходится на сигнатурный анализ, остальные 70% приходятся на метод эвристического анализа и эмуляции исполнения.

Так же, с развитием ЭВМ и появлением нового оборудования, появились аппаратные уязвимости, которые полностью можно было устранить, только заменой оборудования на более новое, зачастую более дорогостоящее.

Примером аппаратной уязвимости является уязвимость Meltdown, позволяющая любой вирусной программе, путём выполнения определённых инструкций извлечь любые данные из кэша процессора. Атака возможна благодаря манипуляции с внеочередным исполнением внутри процессора. Технология внеочередного исполнения команд процессором создана для того, чтобы определить наиболее приоритетные инструкции в конвейере обработки инструкций и выполнить их в первую очередь. Однако этим процессом можно манипулировать таким образом, что можно извлечь любую информацию, начиная с обычной строки, заканчивая ключами шифрования и пользовательскими данными. Уязвимости подвержены все процессоры, которые выпускаются с 1995 года.

Способы противодействия компьютерным вирусам можно разделить на несколько групп:

– профилактика заражения и уменьшение предполагаемого ущерба от такого заражения;

– методика использования антивирусных программ, в том числе обезвреживание и удаление известного вируса;

– способы обнаружения и удаления неизвестного вируса.

Наиболее эффективны в борьбе с компьютерными вирусами антивирусные программные средства. Однако не существует антивирусных средств, гарантирующих полную защиту от вирусов, и заявления о существовании таких систем можно расценить как-либо недобросовестную рекламу, либо непрофессионализм. Таких систем не существует, поскольку на любой алгоритм антивируса всегда можно предложить контр-алгоритм вируса, невидимого для этого антивируса.

Основными признаками заражения вирусом являются:

– замедление работы компьютера;

– невозможность загрузки операционной системы;

– частые зависания и сбои в работе компьютера;

– прекращение работы или неправильная работы ранее функционировавших программ;

– увеличение количества файлов на диске;

– изменение размера файлов;

– периодическое появление на экране монитора неуместных системных сообщений;

– уменьшение объема свободной оперативной памяти;

– заметное возрастание времени доступа к файлам на жестком диске;

– изменение даты и времени создания файлов;

– разрушение файловой структуры (исчезновение файлов, искажение каталогов и др.).

Следует также обратить внимание на несколько терминов, применяемых при обсуждении антивирусных программ:

– ложное срабатывание – детектирование вируса в незараженном объекте (файле, секторе или системной памяти);

– сканирование по запросу – поиск вирусов по запросу пользователя. В этом режиме антивирусная программа неактивна до тех пор, пока не будет вызвана пользователем из командной строки, командного файла или программы-расписания;

– сканирование на лету – постоянная проверка на вирусы объектов, к которым происходит обращение (запуск, открытие, создание). В этом режиме антивирус постоянно активен, он присутствует в памяти «резидентно» и проверяет объекты без запроса пользователя.

Далее приведены основные понятия и ключевые слова, необходимые для понимания предметной области.

Компьютерный вирус – это вид вредоносного программного обеспечения, способного создавать копии самого себя и внедряться в код других программ, системные области памяти, загрузочные секторы, а также распространять свои копии по разнообразным каналам связи

**Антивирусная программа (антивирус)** – изначально компьютерная программа, которая предназначена для обезвреживания вирусов и различного рода вредоносного ПО, с целью сохранности данных и оптимальной работы вашего персонального компьютера.

Сигнатура – непрерывная конечная последовательность байт, необходимая и достаточная для однозначной идентификации угрозы.

Шины устройств – подсистемы передачи данных между функциональными блоками компьютера (например, шина USB, PCI).

Хеш-сумма – уникальный идентификатор файла, представляющий собой последовательность цифр и букв заданной длины. Используется для проверки целостности данных.

Эксплойт – программа, фрагмент кода или последовательность команд, использующие уязвимости в программном обеспечении и применяемые для проведения атаки на систему.

Ядро программного средства – совокупность компонентов и связей между ними.

IRP пакет – пакет запроса ввода-вывода.

SSDT – таблица системных вызовов.

GDT – глобальная дескрипторная таблица.

IDT – таблица векторов прерываний.

LDT – локальная дескрипторная таблица.

1.2 Назначение программного средства

Основное назначение антивирусных программных средств – обеспечение базовой защиты, предотвращение угроз, восстановление системы и данных, защита конфиденциальных данных.

В состав базовой защиты в первую очередь входят функции:

– защиты от вирусов, червей, эксплойтов, троянских, шпионских и рекламных программ;

– проверка файлов, почтовых сообщений, интернет в автоматическом режиме и по требованию;

– мониторинг активности (сбор данных о работе программ на компьютере);

– откат действий вредоносной программы;

– постоянная защита от фишинговых сайтов;

– постоянная проверка файлов в автономном режиме.

В состав функции предотвращения угроз входит:

– поиск уязвимостей в операционной системе и установленном ПО;

– блокирование ссылок на заражённые сайты;

– распознавание вирусов по способу их упаковки;

В состав функции восстановления системы и данных входит:

– возможность упаковки программы на заражённый компьютер;

– функция самозащиты программы от выключения или остановки;

– восстановление корректных настроек системы после удаления вредоносного ПО;

– наличие инструментов для создания диска аварийного восстановления.

В состав функции защита конфиденциальных данных входит:

– блокирование ссылок на фишинговые сайты;

– защита от всех видов кейлоггеров и шпионских программ.

1.3 Анализ существующих разработок

Основными критериями оценки антивирусных программных средств являются:

– стабильность и надежность работы – самый определяющий параметр, даже самый лучший антивирус окажется совершенно бесполезным, если он не сможет нормально функционировать на вашем компьютере, если в результате какого-либо сбоя в работе программы процесс проверки компьютера не пройдёт до конца;

– размер вирусной базы программы, сюда же следует отнести и возможность программы определять разнообразные типы вирусов, и умение работать с файлами различных типов (архивы, документы);

– скорость работы алгоритмов программы, к этому пункту относятся такая характеристика как скорость сканирования или проверки файла;

– кроссплатформенность.

Существует довольно много существующих разработок. Они делятся на платные и бесплатные, но самыми популярными из них, по количеству установок являются: NOD32, Kaspersky, Norton, Doctor Web, McAfee, Avira.

Самыми популярными антивирусными средствами в русскоговорящем сегменте являются Dr Web (доктор веб) и Kaspersky (антивирус Касперского).

В таблице 1.3.1 приведено сравнение антивирусных продуктов.

Таблица 1.3.1 – Сравнение антивирусных продуктов по различным параметрам

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название антивирусной программы | Kaspersky | NOD 32 | Symantec | Dr. Web | 360 Total security |
| Поддержка в системах семейства Windows | Имеется | Имеется | Имеется | Имеется | Имеется |
| Поддержка в системах семейства Linux | Имеется | Не имеется | Не имеется | Имеется | Не имеется |
| Время загрузки системы | До 4 минут | До 3 минут | До 3 минут | До 4 минут | До 2 минут |
| Время сканирования системных папок | Меньше 20 минут | Меньше 10 минут | Меньше 15 минут | Меньше 17 минут | Меньше 10 минут |
| Загрузка ЦП | 5.5 – 8 % | 8%–12% | 4% –10% | 3% – 8% | 2.5% – 5% |
| Использование ОЗУ в простое | 210 мб | 138 мб | 157 мб | 120 мб | 107 мб |
| Антивирусный сканер и монитор | Имеется | Имеется | Имеется | Имеется | Имеется |
| Защита персональных данных | Имеется | Имеется | Имеется  (в платной версии) | Имеется | Имеется |
| Фаервол | Имеется | Имеется | Имеется | Не имеется | Не имеется |
| Защита почтового ящика | Имеется | Имеется | Имеется | Имеется | Не имеется |
| Работа в облаке | Не имеется | Не имеется | Имеется | Не имеется | Имеется |
| Обнаружение вторжения | Имеется | Имеется | Имеется | Имеется | Не имеется |
| Система обновлений | Имеется | Имеется | Имеется | Имеется | Имеется |
| Веб-защита | Имеется | Имеется | Не имеется | Имеется | Имеется |
| Антиспам | Имеется | Имеется | Имеется | Имеется | Не имеется |
| Поддержка большинства языков | Имеется | Имеется | Имеется | Имеется | Имеется |

На основании проведенного сравнения, можно сделать вывод что одним из лучших антивирусных средств, является антивирус Касперского. Именно он обеспечивает наилучшую защиту от взломщиков, троянов и других видов вирусных программ.

2 Технические характеристики

2.1 Постановка задачи на разработку

Целью дипломного проекта является разработка программного средства, позволяющего защитить операционную систему от файловых вирусов.

Для достижения поставленной цели были сформулированы следующие задачи:

– разработать логическую структуру программного средства;

– разработать и протестировать библиотеки, модули и драйвер операционной системы.

Программное средство должно обеспечивать следующий функционал:

– защита операционной системы от файловых вирусов;

– хранение файловых вирусов в защищенном хранилище;

– ручное сканирование;

– настройка исключений;

– автоматическая проверку файлов в фоновом режиме.

2.2 Описание программного средства

Текст

2.2.1 Общие сведения

Текст

2.2.2 Функциональное назначение

Текст

2.2.3 Описание логической структуры

На рисунке 2.2.3.1 изображена логическая структура программного средства.

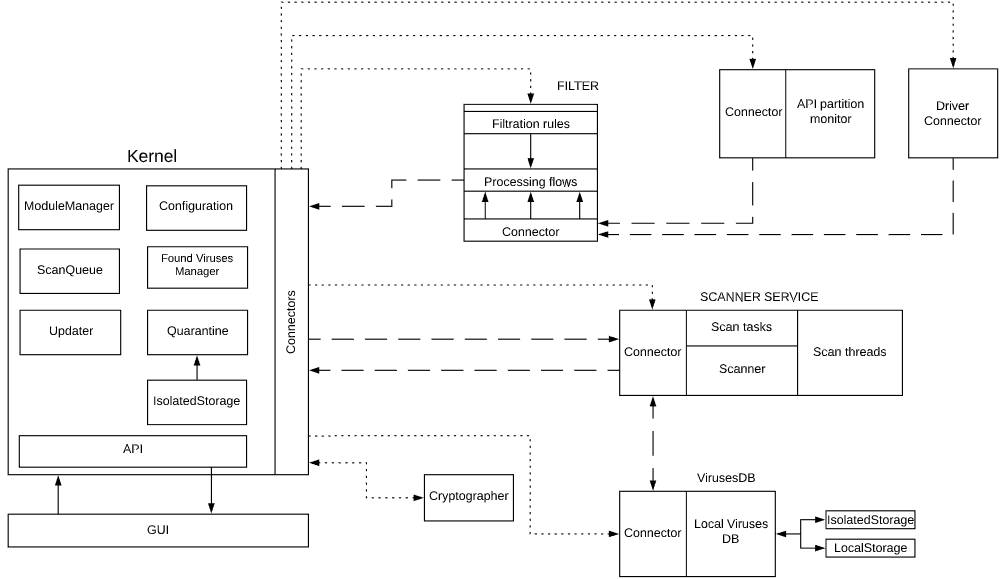


Рисунок 2.2.3.1 – Логическая структура программного средства

В таблице 2.2.3.1 приведено описание всех модулей программного средства

Таблица 2.2.3.1 – описание компонентов и модулей программного средства

|  |  |
| --- | --- |
| Компонент | Описание |
| Kernel | Ядро программного средства, связывает воедино все модули, состоит из 8 компонентов |
| Filter | Модуль фильтрации, позволяет отфильтровать непригодные для сканирования файлы и пути, состоит из 3 компонентов |
| Api partition monitor | Модуль обнаружения создания или изменения файлов на жестком диске |
| Scanner service | Модуль сканирования файлов, состоит из 4 компонентов |
| Viruses DB | Модуль локальной базы сигнатур |
| Cryptographer | Модуль криптографа |
| GUI | Модуль графического интерфейса |
| Driver connector | Модуль коннектора, позволяет соединить ядро программного средства с драйвером ядра операционной системы |

В таблице 2.2.3.2 приведено описание внутренних компонентов ядра программного средства

Таблица 2.2.3.2 – Компоненты ядра программного средства

|  |  |
| --- | --- |
| Компонент | Описание |
| ModuleManager | Менеджер модулей, предназначен для загрузки и управления работой сторонних DLL модулей |
| ScanQueue | Менеджер задач сканирования, хранит в себе активные задачи по сканированию файлов, компонент взаимодействует с сервисом сканирования |
| Updater | Модуль обновления, позволяет получать обновления через интернет |
| Configuration | Компонент конфигурации, содержит в себе настройки позволяющие задавать работу модулей и компонентов |
| Found Viruses Manager | Менеджер обнаруженных вирусов, хранит в себе полную информацию об обнаруженных вирусах |
| Quarantine | Компонент карантина, позволяет хранить обнаруженные вирусы в защищенном хранилище |
| Connectors | Компонент соединений, позволяет другим компонентам ядра взаимодействовать с DLL модулями |
| API | Компонент позволяющий связать ядро со сторонним пользовательским приложением |

Все DLL модули имеют одну схожую черту, все они имеют в своём составе один и тот же компонент, называемый коннектором. Коннектор позволяет связать модуль с другим модулем или приложением с помощью именованных каналов. Именованный канал представляет собой поток байтов, доступ к которому может иметь несколько приложений, данные в такой поток можно записывать и считывать.

При запуске ядра первоначально производится инициализация конфигурации, затем следует загрузка DLL модулей, инициализация входящих и исходящих подключений и только после этого производится инициализация внутренних компонентов ядра. Такой порядок обусловлен тем, что компоненты ядра работают с DLL модулями, чтобы удостоверится в работоспособности модулей, необходимо создать подключение к ним, если подключение успешно, то производится инициализация внутренних компонентов. Процесс полной инициализации изображен на блок-схеме 2.2.3.2.

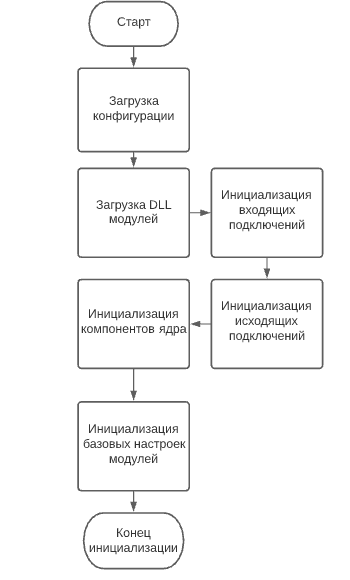


Рисунок 2.2.3.2 – Блок-схема процесса полной инициализации ядра

Менеджер модулей хранит описание и различные данные о DLL модуле. В таблице 2.2.3.3 описаны данные, которые хранит менеджер модулей о каждом модуле.

Таблица 2.2.3.3 – Описание хранимых данных о DLL модуле

|  |  |
| --- | --- |
| Наименование поля | Описание |
| ModuleName | Наименование модуля |
| ModuleAssembly | Ссылка на сборку модуля |
| IsRunning | Состояние модуля |

На рисунке 2.2.3.3 изображена блок-схема процесса загрузки DLL модуля.

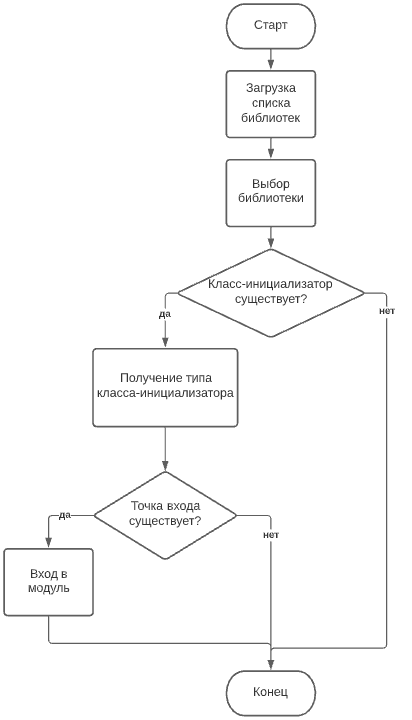


Рисунок 2.2.3.3 – Блок-схема процесса загрузки DLL модуля

Все DLL модули всегда инициализируют подключения к другим модулям. В таблице 2.2.3.4 описаны действия, выполняемые при инициализации каждого модуля.

Таблица 2.2.3.4 – Действия выполняемые при инициализации каждого модуля

|  |  |
| --- | --- |
| Наименование инициализируемого модуля | Описание действий |
| API Partition monitor | Запуск потока выполнения команд |
| Filter | Загрузка стандартных правил фильтрации, запуск потоков-обработчиков |
| Scanner service | Запуск потоков-обработчиков |
| Viruses DB | Запуск потока обработки команд, загрузка файлов базы сигнатур в защищенное хранилище, загрузка сигнатур в ОЗУ из защищенного хранилища |

В таблице 2.2.3.5 описаны действия, выполняемые при инициализации каждого компонента ядра.

Таблица 2.2.3.5 – Действия выполняемые при инициализации каждого компонента ядра

|  |  |
| --- | --- |
| Наименование инициализируемого компонента | Описание действий |
| API | Установка событий, запуск потока-обработчика запросов |
| ScannerResponseHandler | Запуск потока-обработчика |
| ScanTasks | Запуск потока-обработчика, установка событий |
| FoundVirusesManager | Запуск потока-обработчика |
| Quarantine | Установка защищенного хранилища, проверка существования нужных директорий |

После полной инициализации производится базовая настройка подключенных модулей, а именно:

– установка проверяемых разделов в мониторе разделов;

– установка правил фильтрации;

– генерация или загрузка ключей шифрования;

– выгрузка всех сигнатур из модуля базы сигнатур в модуль сканнера.

Для мониторинга файловой системы в операционной системе используется класс FileSystemWatcher, который в свою очередь взаимодействует с Windows API (application programming interface). На рисунке 2.2.3.4 представлена блок-схема работы монитора изменений.

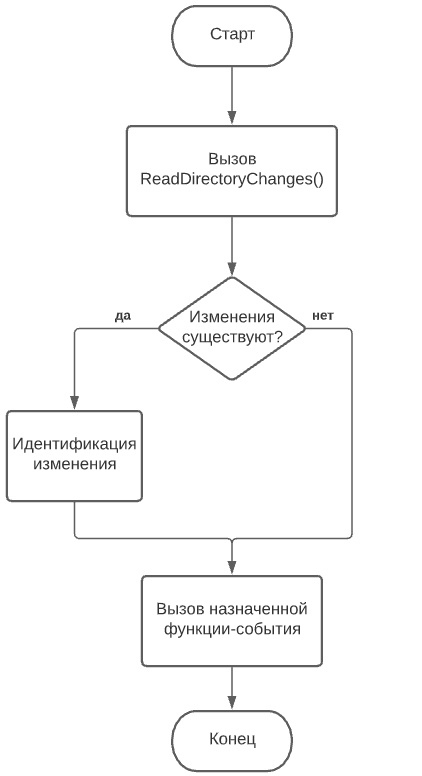


Рисунок 2.2.3.4 – Блок-схема алгоритма монитора изменений

При срабатывании события создания или редактирования файла, путь к файлу и идентификатор действия отправляются в сервис фильтрации.

Сервис фильтрации содержит в себе 3 компонента:

– правила фильтрации;

– коннектор;

– обработчики.

Компонент обработчика представляет собой 3 потока:

– поток обработки команд;

– поток обработки сообщений от модуля монитора разделов;

– поток обработки сообщений от модуля драйвер коннектора.

Существует 3 основных этапа фильтрации и 1 дополнительный. На первом этапе применяются фильтры расширения файлов, на втором фильтры путей, на третьем прочие правила фильтрации, на дополнительном этапе применяются не правила, а функции, в которые передается путь к файлу. Правило фильтрации представляет собой регулярное выражение. Поэтому в правила фильтрации можно добавить только правила, осуществляемые над строкой пути к файлу, а вот с помощью функции можно задать самые различные проверки (например проверка существования файла). На рисунке 2.2.3.4 изображена блок схема процесса фильтрации.

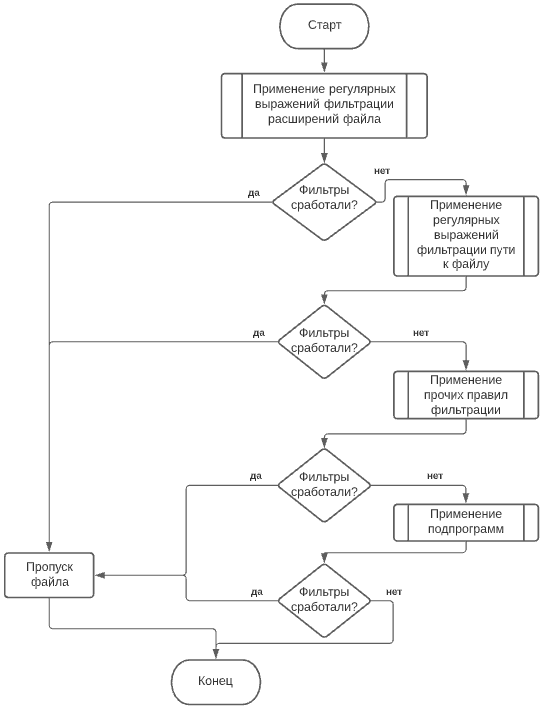


Рисунок 2.2.3.4 – Блок-схема процесса фильтрации

Если какой-либо из фильтров сработал, то файл просто пропускается, если же ни один из фильтров не сработал, то путь к файлу отправляется по именованному каналу дальше в ядро программного средства.

В ядре программного средства обработкой сообщений от модуля фильтра занимается компонент FileQueue, а который содержит в себе специально предназначенный для этого класс FilterHandler.

В таблице 2.2.3.6 приведено описание всех классов, которые содержит в себе компонент FileQueue.

Таблица 2.2.3.6 – Описание классов

|  |  |
| --- | --- |
| Класс | Описание |
| ScannerResponseHandler | Представляет собой обработчик результатов сканирования |
| FilterHandler | Представляет собой обработчик сообщений от фильтра |
| ScanTasks | Представляет собой менеджер активных задач сканирования |
| ScanTask | Представляет собой отдельную задачу сканирования |

При поступлении сообщения с путем к файлу производится проверка на наличие этого файла в листе вирусов, после этого создается задача сканирования. В таблице 2.2.3.6 приведено описание полей класса ScanTask, который представляет собой задачу сканирования.

Таблица 2.2.3.6 – Описание полей класса ScanTask

|  |  |
| --- | --- |
| Наименование поля | Описание поля |
| TaskId | Идентификатор задачи |
| File | Путь к файлу |
| ProbesCount | Количество попыток сканирования |

Все созданные задачи сканирования передаются через именованный канал прямо в модуль сканнера (ScannerService).

Сканнер при приёме задачи сканирования по именованному каналу добавляет её в свой локальный менеджер задач. Все задачи сканирования обрабатываются и выполняются потоками по мере возможности. Количество одновременно сканируемых файлов задается с помощью конфигурации. На рисунке 2.2.3.5 изображена блок-схема алгоритма поиска сигнатуры в файле.

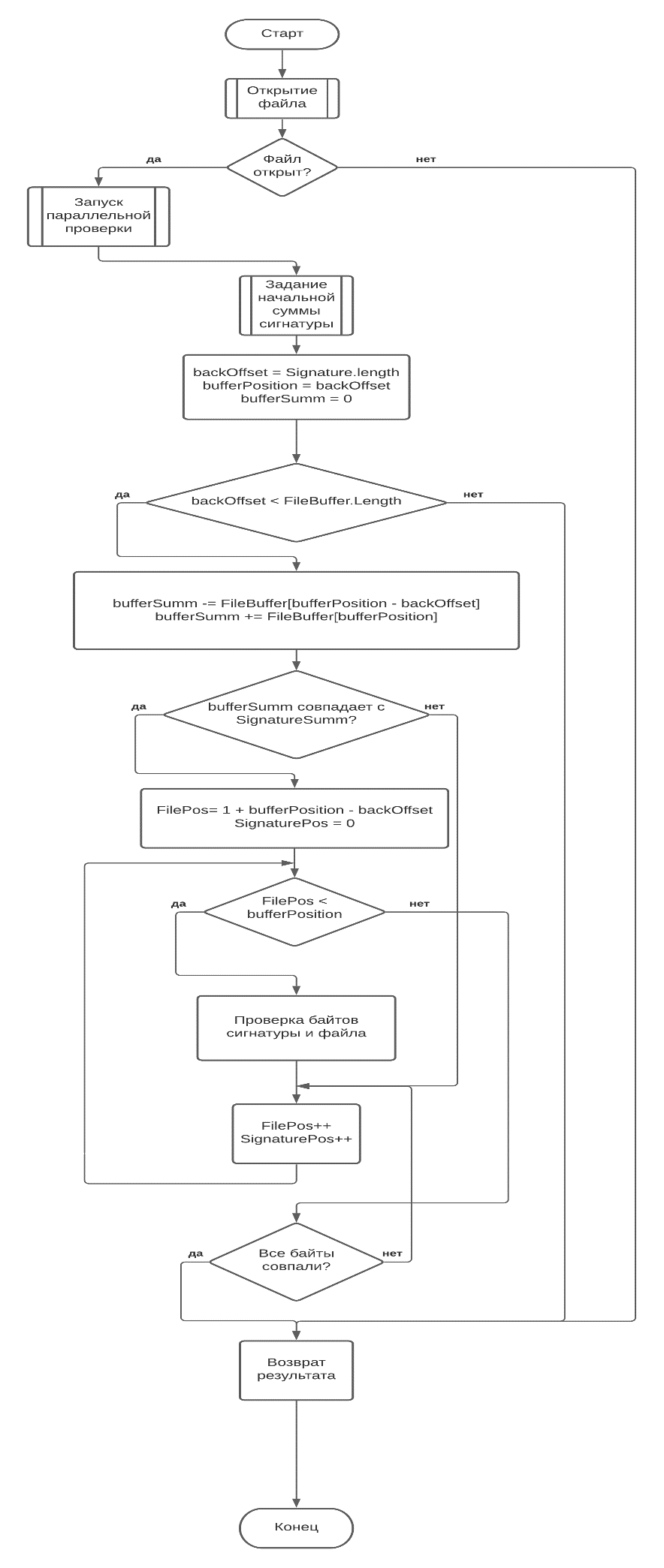


Рисунок 2.2.3.5 – Блок-схема алгоритма поиска сигнатуры

После того как задача сканирования была выполнена, результат сканирования возвращается в компонент FileQueue. Если задача была выполнена с ошибкой, то она повторяется ещё раз. Количество проб зависит от конфигурации компонента. Если задача была выполнена и в файле была обнаружена сигнатура вируса, то файл передается в компонент FoundVirusesManager. Если же в файле не была обнаружена вирусная сигнатура, то задача просто удаляется.

На рисунке 2.2.3.6 приведена блок схема алгоритма проверки файла начиная с создания задачи сканирования и заканчивая её удалением.

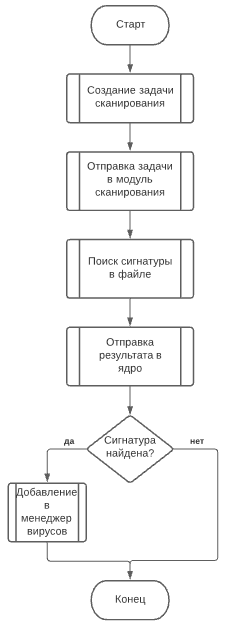


Рисунок 2.2.3.6 – Блок-схема алгоритма проверки файла

Компонент FoundVirusesManager содержит в себе полную информацию о файле и его принадлежности к вирусу. В таблице 2.2.3.7 приведено описание полей класса VirusInfo, который представляет собой информацию о вирусе.

Таблица 2.2.3.7 – Описание полей класса VirusInfo

|  |  |
| --- | --- |
| Наименование поля | Описание поля |
| id | Идентификатор вируса в таблице менеджера |
| inQuarantine | Находится ли вирус в карантине |
| fileInQuarantine | Путь к файлу в карантине |
| file | Путь к файлу на жестком диске |
| VirusId | Идентификатор вирусной сигнатуры |

Если в настройках программного средства включена опция авто перемещения вирусов в карантин, то вирус будет немедленно перемещен в карантин. За реализацию карантина отвечает компонент Quarantine.

Карантин представляет собой изолированное хранилище, доступ к которому может получить только та программа, которая создала это хранилище. В изолированном хранилище данные всегда изолированы по пользователю и по сборке. Учетные данные, такие как происхождение или строгое имя сборки, определяют идентичность сборки. Данные также могут быть изолированы по домену приложения с использованием аналогичных учетных данных. Для того, чтобы файл поместить в карантин, необходимо его сначала скопировать в защищенное хранилище, а затем удалить оригинал на жестком диске.

На рисунке 2.2.3.7 изображена блок-схема алгоритма перемещения файла в карантин.

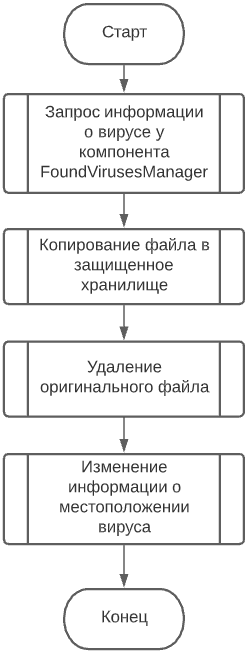


Рисунок 2.2.3.7 – Блок-схема алгоритма перемещения файла в карантин

Что бы другие программные средства могли взаимодействовать с ядром был реализован специальный API. Библиотека API связывается с ядром с помощью именованного канала. Ядро таким образом может принимать команды или уведомлять стороннее программное средство при возникновении различных событий (например событие обнаружение вируса). На рисунке 2.2.3.8 приведена диаграмма последовательности действий работы ядра совместно с графическим интерфейсом пользователя.

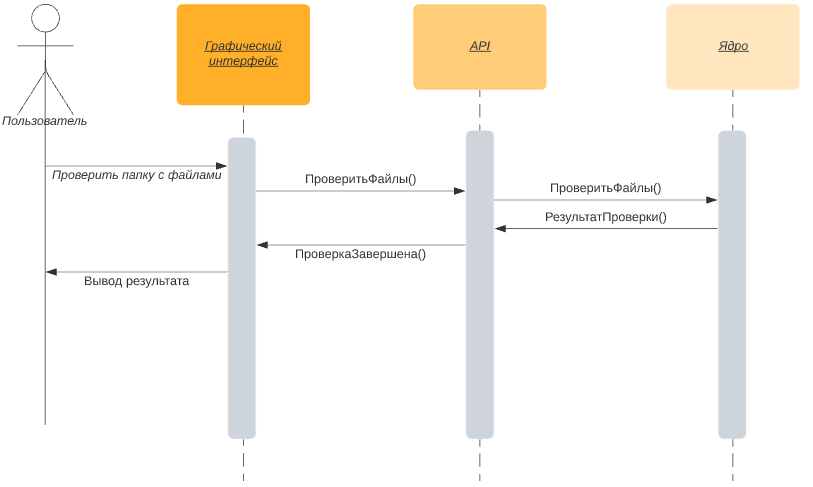


Рисунок 2.2.3.8 – Диаграмма последовательности действий

Для осуществления действий над обнаруженными вирусами требуется вмешательство пользователя. На рисунке 2.2.3.9 изображена блок-схема взаимодействия ядра программного средства с пользователем при обнаружении вируса.



Рисунок 2.2.3.9 – Блок–схема взаимодействия

В таблице 2.2.3.8 приведено описание функционала библиотеки API.

Таблица 2.2.3.8 – Описание функций библиотеки API

|  |  |
| --- | --- |
| Функция | Назначение |
| ToQuarantine | Поместить вирус в карантин |
| RestoreFile | Восстановить файл из карантина |
| GetVirusInfo | Получить информацию о вирусе |
| ApplyingActions | Применить действия к вирусам |
| ClearScanQueue | Очистить очередь сканирования |
| AddToScan | Добавить файл для проверки в очередь сканирования |
| SetAutoScanRemovableDevices | Установить опцию авто сканирования съемных носителей |
| getAllViruses | Получить информацию о всех вирусах |
| DeleteFile | Удалить файл, где бы он не находился (в карантине или на жестком диске) |
| ClearConnectedDevices | Очистить таблицу подключенных устройств |
| AddSimpleRule | Добавить простое правило фильтрации |
| RemoveSimpleRule | Удалить простое правило фильтрации |
| ClearSimpleRules | Удалить все простые правила фильтрации |

В таблице 2.2.3.9 приведено описание событий библиотеки API.

Таблица 2.2.3.9 – Описание событий библиотеки API

|  |  |
| --- | --- |
| Событие | Назначение |
| onScanCompleted | Срабатывает при выполнении сканирования |
| onScanFound | Срабатывает при обнаружении вируса |
| onVirusInfo | Срабатывает при получении информации о вирусе |

2.2.4 Используемые технические средства

В таблице 2.2.4.1 приведены минимальные аппаратно-программные характеристики для компьютера разработчика.

Таблица 2.2.4.1 – Минимальные аппаратно-программные характеристики для компьютера разработчика

|  |  |
| --- | --- |
| Компонент | Требование |
| Процессор | Ryzen 5 2200 |
| Оперативная память | 1 GB DDR4 RAM |
| Жесткий диск | 100MB HDD |
| Операционная система | Windows 8.1 |
| Видеокарта | Vega 8 |

В таблице 2.2.4.2 приведены минимальные аппаратно-программные характеристики для компьютера пользователя.

Таблица 2.2.4.2 – Минимальные аппаратно-программные характеристики для компьютера пользователя

|  |  |
| --- | --- |
| Компонент | Требование |
| Процессор | Ryzen 5 1600 |
| Оперативная память | 1 GB DDR4 RAM |
| Жесткий диск | 50MB HDD |
| Операционная система | Windows 8.1 |
| Видеокарта | Vega 8 |

НУЖНЫ ПРОГРАММНЫЕ ТРЕБОВАНИЯ ЕЩЁ!

2.2.5 Вызов и загрузка

Для обеспечения максимальной безопасности загрузка программного средства производится автоматически при загрузке операционной системы.

2.2.6 Входные данные

Все входные данные в модули передаются с помощью именованных каналов. Именованные каналы представляют односторонние или дуплексные каналы для связи между сервером канала и одним или несколькими клиентами. Именованные каналы могу использоваться для межпроцессного взаимодействия локально или по сети. Любой процесс может действовать как сервер или клиент именованного канала.

Входящие именованные каналы модуля фильтра описаны в таблице 2.2.6.1.

Таблица 2.2.6.1 – Описание входных каналов модуля фильтра

|  |  |
| --- | --- |
| Именованный канал | Назначение |
| Filter.CommandPipe | Канал для команд |
| API\_MON\_FILTER | Канал для приёма сообщений от модуля мониторинга разделов жесткого диска и съемных носителей |
| DRIVER MON | Канал для приёма сообщений от драйвера мониторинга операций ввода-вывода |

Модули мониторинга разделов, криптографа, API и вирусной базы сигнатур в своём составе имеют только 1 именованный канал, предназначенный для приёма команд.

В таблице 2.2.6.2 приведено описание входных именованных каналов модуля сканнера.

Таблица 2.2.6.2 – Описание входных именованных каналов сканнера

|  |  |
| --- | --- |
| Именованный канал | Назначение |
| Scanner.CommandPipe | Канал для команд |
| ScannerService.Input | Канал для приёма путей к сканируемым файлам |
| ScannerService.Signatures | Канал для приёма сигнатур вирусов от модуля базы сигнатур |

2.2.7 Выходные данные

Выходные данные в модулях передаются так же, как и входные ­– с помощью именованных каналов

В таблице 2.2.6.3 приведено описание выходных именованных каналов.

Таблица 2.2.6.3 – Описание выходных именованных каналов

|  |  |  |
| --- | --- | --- |
| Наименование модуля | Наименование канала | Назначение |
| Filter | Filter.Output | Канал для передачи файлов для проверки |
| ScannerService | ScannerService.Output | Канал для передачи проверенных файлов |
| Cryptographer | Crypto.Data | Канал для передачи различных данных |
| API | API.User | Канал для передачи данных событий |
| API Monitor | API MON FILTER | Канал для отправки путей к файлам |
| VirusesDB | ScannerService.Signatures | Канал для передачи сигнатур в модуль сканнера |

2.2.8 Рекомендации по использованию