

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

СИСТЕМА ОБЕЗВРЕЖИВАНИЯ ВРЕДОНОСНЫХ
ПРОГРАММ НЕСАНКЦИОНИРОВАННОГО
ДОСТУПА К КРИПТОКОШЕЛЬКАМ

Студент

Немцев 31.05.22 Немцев А.А.
(подпись, дата)

Руководитель

Зинкин 31.05.22 Зинкин С.А.
(подпись, дата)

Нормоконтролёр

А.С. Бычков 31.05.22 А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол зав. кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой

М.А. Митрохин
(подпись)

М.А. Митрохин

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

20 июня 2022 № 16)

Секретарь ГЭК

Е.И. Гурин
(подпись)

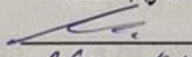
Е.И. Гурин

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«26» 04 2022 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Немцев Александр Андреевич гр. 18вв1 факультета вычислительной техники.
2. Тема работы «Система обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам».

Тема утверждена приказом ПГУ № 357/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы д.т.н., доцент каф. ВТ ПГУ Зинкин С.А.

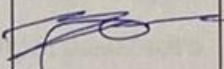
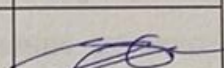
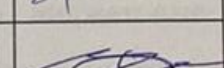
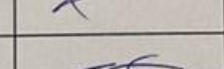
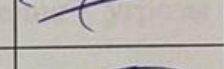
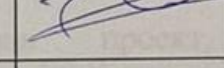
4. Задание на работу (назначение разработки, исходные данные и т.п.).

Назначение разработки – систему обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам. Приложение должно обнаруживать и устранять угрозы ВПО направленного на виджетовые криптокошельки, а также обладать графическим и интуитивно понятным интерфейсом.

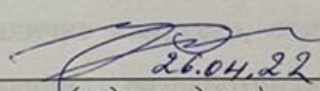
5. Перечень подлежащих разработке вопросов.

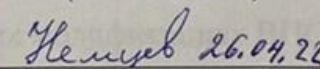
Требуется произвести анализ предметной области: изучить аналоги программ со схожей функциональностью, выделить достоинства и недостатки. Определить архитектуру приложения и средства разработки. Выбрать язык программирования и доступные библиотеки

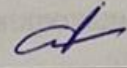
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2022 – 28.05.2022	
2	Анализ требований	32	29.04.2022 – 30.05.2022	
3	Создание макета графического интерфейса	30	1.04.2022 – 3.05.2022	
4	Выбор технологий, применяемых при проектировании и разработке	40	4.05.2022 – 8.05.2022	
5	Реализация графического интерфейса	60	9.05.2022 – 15.05.2022	
6	Разработка основной части программы	76	16.05.2022 – 23.05.2022	
7	Оформление пояснительной записки	54	24.05.2022 – 31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы  26.04.22 Зинкин С.А.
(подпись, дата)

Задание к исполнению принял студент  26.04.22 Немцев А.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  16.06.2022 Фионова Л.Р.
(подпись, дат)

Реферат

Работа содержит 60 листов, 13 рисунков, 3 приложения.
СИСТЕМА ОБЕЗВРЕЖИВАНИЯ, *PYTHON*, *TELEGRAM API*, *UML*,
SUBLIME TEXT, *PYQT*.

В рамках данной выпускной квалификационной работы была создана программа по защите от определенного вредоносного программного обеспечения несанкционированного доступа к криптоактивам.

Цель работы – создать систему обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам. Приложение должно обнаруживать и устранять угрозы ВПО направленного на виджетовые криптокошельки, а также обладать графическим и интуитивно понятным интерфейсом.

Объектом разработки является программа устранения угрозы несанкционированного доступа к криптокошелькам.

В результате проведенной работы реализован проект, удовлетворяющий поставленной задаче, а именно: удобный пользовательский интерфейс, обнаружение вредоносного ПО несанкционированного доступа к криптокошелькам, устранение найденных угрозы, сканирование файлы на наличие «вируса», отправка информации пользователем для информирования разработчика о возможных недоработках программы или о новых модификациях ВПО.

Разработка проводилась в среде программирования *Sublime Text*.
Использованные языки программирования: *Python*.

Созданная программа не имеет аналогов, так как выполняет задачу по защите от конкретного, но достаточно распространенного вируса, который нацелен на пользователей с виджетовыми криптокошельками.

					ПГУ 1.090301.28.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Система обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Немцев А.А.					2	57	
Проверил		Зинкин С.А.							
Н. Контр.		Бычков А.С.							
Утверд.		Митрохин М.А.							
						ФВТ, каф. ВТ Гр. 18ВВ1			

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Система обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам	9
1.1 Анализ предметной области	9
1.1.1 Анализ конкурентов	10
1.2 Постановка задачи	12
1.3 Архитектура приложения	13
1.4 Инструменты разработки.....	14
1.4.1 Выбор языка разработки	14
1.4.2 Выбор среды разработки.....	17
1.4.3 Выбор платформы для реализации графического интерфейса.....	18
1.5 Реализация	19
1.5.1 Модуль <i>Clear</i>	19
1.5.2 Модуль <i>Scan</i>	20
1.5.1 Модуль <i>Feedback</i>	22
1.6 Описание работы программы	23
2 Охрана труда.....	31
2.1 Анализ негативных факторов, воздействующих на человека	31
2.1.1 Эргономика рабочего места.....	31
2.1.2 Правила перерывов и режима работы.....	32
2.1.3 Электробезопасность	32
2.1.4 Освещенность рабочего места	33
Заключение	35
Список использованных источников	36

Приложение А – <i>UML</i> -диаграммы	37
Приложение Б – Листинг программы	42
Приложение В - Презентация	53

					ПГУ 1.090301.28.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

Перечень сокращений и обозначений

API (Application Programming Interface) - интерфейс прикладного программирования, который представляет собой набор определений и протоколов для создания и интеграции прикладного программного обеспечения.

NFT (Non Fungible Token) – невзаимозаменяемый токен, чаще всего представляет собой медиафайл.

UML (Unified Modeling Language) – унифицированный язык моделирования детектов (обнаружений антивирусами)

Детект - обнаружение антивирусами или иными программами

Скам (от англ. scam) - обман, жульничество, мошенничество, надувательство

Стаб - фрагмент программного кода, используемый для конвертации параметров, передаваемых при помощи удаленных вызовов процедур

Стиллер (от английского *to steal*, воровать) – определенный класс вредоносного программного обеспечения, который нацелен на получение нужных данных с устройства «жертвы».

Фреймворк – программная платформа, которая упрощает разработку программного продукта, определяет структуру проекта

Введение

Блокчейн или англ. Blockchain состоит из слияния двух английских слов: Block(блок) и chain(цепь). Что означает выстроенную по определенным правилам цепочку блоков, которые содержат в себе некоторую информацию. Эта информация является хеш суммой данного блока и хеш сумму предыдущего блока, что и обеспечивает связь между блоками. Чтобы соответствовать правилам построения цепочки, изменения хеш суммы нужно будет записать в следующий блок, что вызовет изменения уже его собственной хеш суммы. При этом предыдущие блоки не затрагиваются. Если изменяемый блок является последним в цепочке, то внесение изменений является возможным. Но если после изменяемого блока уже сформировано продолжение, то изменение будет практически невозможным, потому что копия всего блокчейна сохранена на устройствах каждого пользователя этой сети. Такой принцип обеспечивает прозрачность и надежность всей технологии блокчейн.

Само появление понятия «блокчейн» не имеет отношения к какой-либо стране, у него сетевое «гражданство». Впервые данный термин появился в 2008 году. Некоторый анонимный пользователь (а возможно и группа лиц) под псевдонимом Сатоши Накамото опубликовал статью, описывающую характеристики и возможности создания инновационной системы денежных расчетов, основанную на децентрализации.

Биткоин — исторически первое и наиболее известное применение блокчейн-технологии. Транзакции в нем — это переводы средств между анонимными кошельками пользователей. Каждому участнику доступна информация о любой из транзакций, когда-либо происходивших в блокчейне, начиная с первого перевода в 2009 году.

Технология блокчейн может быть полезна не только для совершения транзакций. Практически сразу после появления биткоина, люди начали думать, как применить эту технологию в других сферах. Когда майнеры

					ПГУ 1.090301.28.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

проверяют транзакции, они запускают небольшие программы, которые обрабатывают и предоставляют необходимые для транзакции данные. Но технологию можно также использовать для запуска более сложных программ, например, программное обеспечение для социальных сетей. Или использовать блокчейн для предоставления данных для онлайн-форумов.

Эти идеи появились сразу после создания биткойна, но только в 2013 году Виталик Бутерин, на тот момент 19-ти летний студент из Торонто, внёс ощутимый вклад в их развитие. Он разработал совершенно новую технологию под названием *Ethereum*. Благодаря ей блокчейн можно стало использовать не только для совершения транзакций.

В отличие от биткойна, *Ethereum* использует мини-программы, так называемые смарт-контракты, которые пишутся на специальном языке программирования *Solidity*. Пользователи могут взаимодействовать с программами, отправляя им транзакции с инструкциями, которые затем обрабатывают майнеры.

На базе *Ethereum* были созданы и первые *NFT*, которые так сильно набирают популярность в последнее время. *NFT* — это non-fungible token, невзаимозаменяемый, или уникальный токен. Они представляют собой в основном арт-объекты или другие виды цифрового искусства.

Это происходит на специальных платформах, работающих с *NFT*. Есть несколько популярных площадок — например, *Rarible*, *Mintable*, *OpenSea*. Последняя — самая крупная. Все они являются одновременно *NFT*-маркетплейсами и *NFT*-мастерскими.

Сам по себе блокчейн фактически является реестром записей. Например, биткойн или эфириум — записи в блокчейне. *NFT* — тоже. Такие токены, как и любую криптовалюту, можно хранить в своем криптокошельке и совершать с ними транзакции.

Самые популярные кошельки на данный момент это *Phantom* и *Metamask*. Для создания *NFT*-токена понадобится кошелек в системе *Ethereum*

					ПГУ 1.090301.28.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

или *Solana*, а также файл с цифровым контентом. Обычно алгоритм действий похож на работу с виртуальным фотоальбомом. Просто создается коллекция и добавляется в нее медиафайл (или файлы) в любом из доступных форматов — от PNG до MP4. Далее выставляется цена продажи или аукцион и *NFT* размещено на площадке.

На данном этапе доверчивым и амбициозным авторам цифрового искусства могут приходить предложения по их продвижению или покупке их *NFT*, но при условии скачивания какого-нибудь файла. Такой вредоносный файл злоумышленник отправляет конкретному автору с расчётом на то, что у него имеется неплохой капитал криптовалюты на кошельке. И если создатель *NFT* не будет достаточно бдительным и его получится ввести в заблуждение с помощью социальной инженерии, то его криптоактивами завладеют мошенники.

					ПГУ 1.090301.28.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 Система обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам

1.1 Анализ предметной области

С развитием новых технологий, а тем более с появлением новых цифровых валют появляются и новые способы мошенничества. Цифровые активы, хранящиеся на столь удобных и популярных виджетовых кошельках, являются главной целью киберпреступников. И в основном они нацелены на такие известные кошельки как «*Phantom*» или «*Metamask*».

Злоумышленники разными способами пытаются получить доступ к цифровым активам, используя уже привычные методы по типу «фишинга», а также создают новые вариации «стиллеров» под самые популярные криптокошельки.

Стиллер (от английского *to steal*, воровать) – определенный класс вредоносного программного обеспечения, который нацелен на получение нужных данных с устройства «жертвы». Такими данными в основном являются: логины, пароли, файлы *cookies*, *keylog*, скриншоты экрана, логи буфера обмена и прочие данные сохраненные в системе. При скачивании стиллера, с устройства «жертвы» отправляются данные на устройство злоумышленника, где они обрабатываются и используются в целях несанкционированного доступа к аккаунтам и персональным данным [1].

Производится массовая рассылка пользователям, у которых возможно имеются криптовалюты на виджетовых кошельках. В большинстве случаев мошенники с помощью социальной инженерии входят в доверие к жертве и в таком случае потеря средств лежит полностью на плечах человека, который под тем или иным предлогом решил скачать незнакомый исполняемый файл.

Конечно же, не будет лишним добавить тот факт, что антивирус не «видит» зараженный файл. Даже когда «стиллер» начинает свою работу ни один из общеизвестных антивирусов не может засечь подозрительную

активность, ведь киберпреступники адаптировались к современным программа защиты и используют простые, но незаметные алгоритмы для получения нужной им информации.

Будучи вовлеченным в изучение инновационной технологии блокчейн и соответственно принципом работы крипторынка, наткнулся на проблему вредоносного ПО, которое распространялось в основном на держателей средств в наиболее популярных кошельках *Phantom* и *Metamask*.

Углубившись в эту тему, я нашел на некотором теневом форуме информацию о наборе людей в преступную группу по «скаму» криптокошельков. Скам (от англ. *scam*) - обман, жульничество, мошенничество, надувательство. Значит именно в таком сообществе мне удастся получить информацию про «стиллер», чтобы узнать какими методами можно от него защититься. Представившись опытным программистом, у которого есть опыт написания собственных вирусов, я начал вести общение с лидером преступного сообщества с целью получить исходный код вредоносного ПО. Расспрашивая о принципе работы вредоносной программы, а также предлагая способы его улучшения, через некоторое время я получил код и начал составлять алгоритм действий направленных на защиту криптокошельков от несанкционированного доступа[2].

1.1.1 Анализ конкурентов

В данном случае, конкурирующими программами можно назвать разве что самые популярные антивирусы на данный момент. Антивирусная программа — специализированная программа для обнаружения компьютерных вирусов, а также нежелательных (считающихся вредоносными) программ и восстановления заражённых (модифицированных) такими программами файлов и профилактики — предотвращения заражения (модификации) файлов или операционной системы вредоносным кодом. В данном разделе будет рассмотрено 3 программы которые могли бы обнаруживать вредоносный файл передачи персональной информации.

					ПГУ 1.090301.28.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

Антивирус Касперского - Антивирусное программное обеспечение, разрабатываемое Лабораторией Касперского. Предоставляет пользователю защиту от вирусов, троянских программ, шпионских программ, руткитов, adware, а также от неизвестных компоненту «файловый антивирус» угроз с помощью проактивной защиты.

На основе статистики, полученной на форумах, посвященных взломам компьютерной информации, можно заверить, что данный продукт проигрывает своим конкурентам в скорости, а главное в точности определения компьютерных угроз. Антивирус может считать лицензированный продукт вредоносным. Да недавнего времени антивирус не делал упор на проверку цифровой подписи приложений, в следствии чего программы с вредоносным кодом и скомпрометированным цифровым сертификатом имели огромные возможности для реализации эксплуатаций на устройстве “жертвы”.

Защитник *Windows* - Программный продукт компании *Microsoft*, созданный для того, чтобы удалять, помещать в карантин или предотвращать появление *spyware*-модулей в операционных системах

Microsoft Windows. Windows Defender по умолчанию встроен в операционные системы *Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows 8, Windows 10* и *Windows 11*.

В начале 2020 года антивирус имел уязвимость с связке со встроенной в *Windows 10* групповой политикой данных. Запуск из консоли скрипта позволял без *root*-прав полностью отключить антивирус, либо добавить пусть до установки последующего загруженного инсталлятором файла в исключенные папки для проверки защитным средством. В это-же время параллельно существовала уязвимость с использованием встроенного реестра и того-же редактора групповых политик.

Avast Antivirus - Семейство антивирусных программ, разработанных компанией *Avast* для операционных систем *Windows, Mac OS, Android* и *iOS*. Включает в себя бесплатную и коммерческие версии продуктов, нацеленных

					ПГУ 1.090301.28.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

на компьютерную безопасность и защиту от вредоносных, шпионских и троянских программ, а также остальных типов киберугроз, включая *adware*, червей и фишинг.

Данное средство обладает одним из худших алгоритмов эвристического анализа, сканирование файла проходит за считанные секунды, эвристика не работает на всей дистанции работы файла, вредоносный код, исполняемый с задержкой в 1 минуту, обходит сканирование в песочнице антивируса. Обфускация кода и чистка стаба с поддельной цифровой подписью файла открывают все входы в удалённую систему.

Каждое из рассмотренных выше средств обладает своими нюансами и работает по разным алгоритмам, но все они не решают проблему современного ВПО, которое уже имеет трендовость распространения в определенных кругах владельцев криптовалют. И если задача защитить цифровые активы является первостепенной, то необходима более совершенная система, которая сможет быть приспособлена к современным реалиям мошеннических действий.

1.2 Постановка задачи

Разработать программу по выявлению и уничтожению вредоносного ПО, нацеленного на получение несанкционированного доступа к цифровым активам для основных виджетовых кошельков. Так как вредоносная программа с её настоящими модификациями известна, то необходимо учитывать её исходные данные при написании «защитной» программы, которая должна включать в себя:

- обнаружение вредоносного ПО несанкционированного доступа к криптокошелькам;
- способы устранения найденных угроз;
- удобный и интуитивно понятный пользовательский интерфейс;
- возможность сканирования файлов на наличие «вируса»;

- обратная связь с пользователем, для получения информации о возможных недоработках программы и о новых модификациях ВПО.

1.3 Архитектура приложения

В данном случае программа не будет обладать какой-то шаблонизированной архитектурой, ведь её целью является защита от конкретных действий узконаправленного вредоносного программного обеспечения.

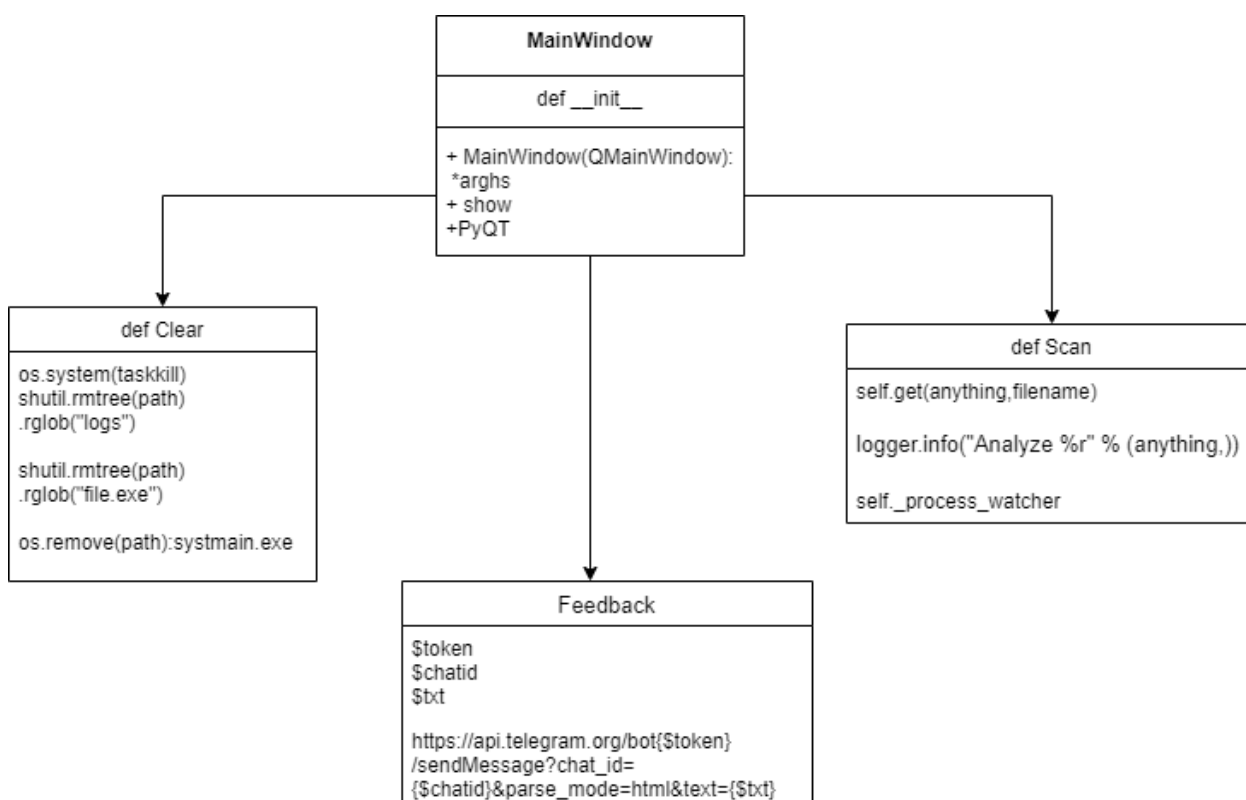


Рисунок 1.1 – UML диаграмма классов

Приложение системы защиты криптокошелька состоит из следующих основных модулей:

- *MainWindow* – основное окно графического интерфейса *PyQT*;
- *Scan* - производится анализ на основе проведенных тестов сразу у огромной базы антивирусов, попадает на виртуальные машины и проводятся более углубленные тесты, в результате которых плохо закриптованные вирусы

с известным стабом (стабом в распределённых вычислениях понимается фрагмент программного кода, используемый для конвертации параметров, передаваемых при помощи удаленных вызовов процедур) быстро наберут детектов (обнаружений антивирусами);

- *Clear* - проводится проверка на изменение внутри определенных директорий. При обнаружении подозрительной активности, происходит моментальное удаление несанкционированных файлов и завершается процесс подмены файлов;

- *Feedback* - Обратная связь, где пользователи смогут сообщать о новых модификациях ВПО посредством отправки сообщения через подключенный API мессенджера *Telegram*.

1.4 Инструменты разработки

1.4.1 Выбор языка разработки

При выборе языка программирования для создания системы безопасности криптокошелька стоит руководствоваться актуальностью, простотой и функционалом. Для выбора подходящего языка стоит сравнить самые популярные на данный момент языки программирования и выбрать лучший вариант.

C++ — компилируемый, статически типизированный язык программирования общего назначения, на котором можно создавать программы любого уровня сложности. C++ – это мультипарадигмальный язык (от слова парадигма – стиль написания компьютерных программ), включающий широкий спектр различных стилей и технологий программирования. Часто его причисляют к объектно-ориентированным языкам, но, строго говоря, это не так. В процессе работы разработчик получает абсолютную свободу в выборе инструментов для того, чтобы задача, решаемая с помощью того или иного подхода, была решена максимально эффективно. Иными словами, C++ не понуждает программиста придерживаться только

одного стиля разработки программы (например, объектно-ориентированного)[2].

Синтаксис C++ унаследован от языка Си. Одним из принципов разработки было сохранение совместимости с Си. Тем не менее, C++ не является в строгом смысле надмножеством Си. Со временем, практическая совместимость между языками Си и C++ постепенно будет утрачиваться, так как языки разрабатывают разные группы по стандартизации, не взаимодействующие друг с другом[3].

К главным достоинствам можно отнести:

- высокая вычислительная производительность;
- возможность расширения языка;
- наличие шаблонов, которые дают возможность построения обобщенных алгоритмов и контейнеров.

Недостатками будут являться:

- сложность и избыточность, из-за которых C++ трудно изучать;
- в языке практически полностью сохранён набор конструкций Си;
- шаблоны в своём исходном виде приводят к порождению кода очень большого объёма;
- синтаксис строгий и «многословный»: код читается хуже, чем в некоторых других языках (например в *Python*).

Java — язык программирования общего назначения. Относится к объектно-ориентированным языкам программирования, к языкам с сильной типизацией. Строгая (сильная) типизация не позволяет смешивать в выражениях разные типы и не выполняет автоматически неявные преобразования. Это добавляет дополнительных трудностей: какие-то части приходится прописывать самому, а свободы у программиста меньше, — зато в обмен на это получаете надёжность[4].

У всех качеств *Java*, будь то строгая типизация или объектная ориентированность, есть свои плюсы и минусы, а ещё они есть у самой *Java* как у языка.

Плюсы:

- независимость — код будет работать на любой платформе, которая поддерживает *Java*;
- надёжность — в немалой мере достигается благодаря строгой статической типизации;
- мультифункциональность;
- сравнительно простой синтаксис.

Минусы:

- низкая скорость (по сравнению с Си и C++);
- требует много памяти;
- нет поддержки низкоуровневого программирования (*Java* — высокоуровневый язык). Например, у неё нет указателей.

Но все-таки, для цели создания защитной программы лучше всего подойдет *Python*. Так как приходится ориентироваться под конкретную задачу, то требуется учитывать то, что ВПО написано на языке *Python* и соответственно «защищаться» будет проще когда известно по какому принципу выполняются действия.

Python - это высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Также *Python* - это скриптовый язык программирования. Он универсален, поэтому подходит для решения разнообразных задач и многих платформ, начиная с *iOS* и *Android* и заканчивая серверными ОС [5].

Его преимуществом является то, что это интерпретируемый язык программирования, это означает, что как разработчик вы пишете файлы *Python.py* в текстовом редакторе, а затем помещаете эти файлы в интерпретатор *Python* для выполнения.

Программировать можно практически на всех платформах, язык хорошо спроектирован и логичен. Разработка идёт в разы быстрее, потому что кода здесь куда меньше, чем на других языках.

К плюсам можно отнести:

- простой минималистичный синтаксис: код легко писать, читать и поддерживать;
- динамическая типизация;
- большой выбор фреймворков;
- кроссплатформенность и поддержка почти всех современных систем;
- большая стандартная библиотека и много дополнительных библиотек.

Минусы:

- низкая скорость;
- нагрузка на память;
- сложность при компиляции конечной программы.

1.4.2 Выбор среды разработки

В качестве среды разработки было решено выбрать *Sublime Text*.

Sublime Text — проприетарный текстовый редактор. Редактор работает с несколькими языками программирования. Поддерживает плагины на языке *Python*. *Sublime Text* тонко настраивается и дополняется пакетами для отладки, автозавершения кода, линтинга и т.д. Пристальный взгляд изнутри показывает мощный текстовый редактор, в который загружены плагины, умные функции автозаполнения и многое другое.

У *Sublime Text* очень приятная полоса прокрутки с превью кода, и, конечно же, возможность редактировать текст при помощи нескольких курсоров. Еще из фишек есть подсветка и автокомплит практически чего

удовно, отличный поиск по проекту, что позволяет найти нужную строку или файл всего по нескольким буквам, автоматический перенос слов по заданной ширине строки, проверка грамматики, поддержка разных кодировок и переносов строк, настраиваемая ширина отступов — в общем-то, все, чтобы было комфортно и удобно работать. Еще к плюсам можно отнести гибкую настройку.

К минусам редактора можно отнести проблемы с обратной совместимостью и отсутствие встроенной консоли.

Некоторые возможности:

- быстрая навигация (*Goto Anything*);
- командная палитра (*Command Palette*);
- API плагинов на *Python*;
- одновременное редактирование (*Split Editing*);
- высокая степень настраиваемости (*Customize Anything*).

1.4.3 Выбор платформы для реализации графического интерфейса

PyQT представляет собой набор инструментов виджетов для графического пользовательского интерфейса (GUI). Он был извлечен из библиотеки *Qt*. Является продуктом сочетания языка *Python* и библиотеки *Qt*. Он идет вместе с *Qt Builder*. [5] Его стоит использовать, чтобы получить код *Python* от *Qt*. При поддержке дизайнера *qt* можно создать графический интерфейс, а затем получить код *Python* для этого графического интерфейса. *PyQT* поддерживается на всех платформах: *Windows*, *macOS*, *UNIX*. *PyQT* может использоваться для создания стильных графических интерфейсов.

Преимущества использования:

- универсальность программирования. Программирование графического пользовательского интерфейса с построено на идее сигналов и слотов для создания контакта между объектами. Это позволяет гибко решать инциденты с графическим пользовательским интерфейсом, что приводит к более гладкой кодовой базе;

- больше, чем просто фреймворк: *Qt* использует широкий спектр *API*-интерфейсов собственной платформы для сетевой работы, разработки баз данных и многое другое. Он обеспечивает первичный доступ к ним через специальный *API*;

- различные компоненты пользовательского интерфейса: *Qt* предоставляет несколько виджетов, таких как кнопки или меню, все они разработаны с базовым интерфейсом для всех совместимых платформ. Поскольку *PyQT* является одной из наиболее часто используемых систем пользовательского интерфейса для *Python*, вы можете легко получить доступ к большому количеству документации на различных учебных ресурсах.

Недостатки использования *PyQt*:

- отсутствие документации, специфичной для классов в *PyQt5*;
- требуется много времени, чтобы понять все функции *PyQt*, что означает довольно долгий процесс по получению новых навыков.

1.5 Реализация

Функциональная составляющая программы поделена на 3 основных модуля, выполняющие свои определенные задачи по обеспечению безопасности средств на криптокошельках.

1.5.1 Модуль *Clear*

Первым и самым главным является модуль *Clear* с одноименной функцией. Эта функция проводит проверку внутри определенных директорий, которые могут быть «заражены». Данные директории известны благодаря исходному коду «стиллера». При обнаружении вирусных файлов, программа осуществляет их немедленное удаление.

```
def Clear():  
    time.sleep(10)  
    os.system("taskkill /im 123.exe")
```

```

        try:
shutil.rmtree((next(Path('C:/ProgramData/MicrosoftStorage/').rglob("logs"))))

        except (StopIteration, PermissionError): pass

        try:
            shutil.rmtree((next(Path('C:/ProgramData/Disc-
Soft/').rglob("logs"))))

        except (StopIteration, PermissionError): pass

        try:
shutil.rmtree((next(Path('C:/ProgramData/MicrosoftDefender/').rglob("logs"))))

        except (StopIteration, PermissionError): pass

        try:
shutil.rmtree((next(Path('C:/ProgramData/PackageCache/').rglob("logs"))))

        except (StopIteration, PermissionError): pass

        try:
shutil.rmtree((next(Path('C:/ProgramData/Adobe/Installer/').rglob("logs"))))

        except (StopIteration, PermissionError): pass

        try:
shutil.rmtree((next(Path(r'C://Users/%s/AppData/Roaming/Microsoft/Windows/Start
Menu/Programs/Startup/'
%login).rglob("123.exe"))))

        except (StopIteration, PermissionError): pass

        try:
os.remove(r"C:/Users/%s/AppData/Roaming/Microsoft/Windows/Start
Menu/Programs/Startup/systmain.exe" %login)

        except FileNotFoundError: pass

```

1.5.2 Модуль Scan

При сканировании реализуется функция *Scan*. Её работа состоит в анализе на основе проведенных тестов сразу у огромной базы антивирусов. Файл попадает на виртуальные машины и проводятся более углубленные тесты, в результате которых плохо закриптованные вирусы с известным стабом (стабом в распределённых вычислениях понимается фрагмент программного кода, используемый для конвертации параметров,

					ПГУ 1.090301.28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

передаваемых при помощи удаленных вызовов процедур) быстро наберут детектов(обнаружений антивирусами).

```
        params    =  {'APIkey':  self.API_key,  'resource':
self.checksum}
        reanalyze = reanalyze if reanalyze is not None else
False

        if not reanalyze:

            # Check if already exists

            report = self.get(anything, filename)

            if report is not None:

                return report

        logger.info("Analyze %r" % (anything, ))

        o = self._fileobj_to_fcontent(anything, filename)
        assert o[0] == "file"

        if o[1] is None:

            o[1] = "file"

        if len(o[2]) > FILE_SIZE_LIMIT:

            raise self.EntityTooLarge()

        self._limit_call_handler()

        ret_json = postfile.post_multipart(

            host = "www.virustotal.com",

            selector                                     =
"https://www.virustotal.com/vtAPI/v2/file/scan",

            fields = {

                "APIkey": self.API_key,

            }.items(),

            files = [o,],

        )

        # try except to prevent generic error message and
        provide a more descriptive message

        try:

            response = get(self.url, params=params)

            self.response = response.json()

        except RequestException:
```

```

        exit(1)

    except ValueError:

        exit(1)

    if "scans" not in self.response:

        print_bar("Угроза не обнаружена",
self.response["verbose_msg"], 5)

        exit(1)

        print_bar("Угроза обнаружена!",

                    "Прогресс: %progress " +
str(self.response["positives"]) + "/" +
str(self.response["total"]) +

                    "\nFull report written to
C:\\Users\\{}\\Scan report of {}.txt".format(USERNAME,
self.program_name), 10)

```

Следующий этап сканирования заключается в запуске проверяемого файла и отслеживании в одном потоке моментов создания подозрительных процессов, а в другом – их мгновенного уничтожения.

```

# Отслеживаем события создания процессов
mon_sozdanie = Monitor('sozdanie')

mon_sozdanie.start()

# Отслеживаем события уничтожения процессов
mon_destroy = Monitor('destroy')

mon_destroy.start()

```

1.5.1 Модуль *Feedback*

Функция *Feedback* отвечает за отправку сообщений пользователем. Она реализована посредством интеграции *API* мессенджера *Telegram*.

Отправка сообщение в нужный канал связи осуществляется за счёт выполнения *HTTP GET* запроса к *Telegram BOT API* по следующему *URL*:

```

https://API.telegram.org/bot[BOT_API_KEY]/sendMessage?chat_id=[MY_CHANNEL_NAME]&text=[MY_MESSAGE_TEXT]

```

Сообщения от пользователей приходят в канал разработчика за счёт параметров:

- *token* – токен, который создает *BotFather* для бота, которому нужно присвоить права модератора в канале;

- *channel_id* – идентификатор канала, в который и будут приходить сообщения. Его можно получить с помощью HTTP запроса:

```
https://API.telegram.org/'botname'/getUpdates
```

Или же с помощью бота *@getmyid_bot*, которому можно просто переслать сообщение из нужного чата и он выдаст всю информацию.

```
def feedback(text: str):
    token =
    "'5424533445:AAGzn0F7IyFKjRuFTV3BS5F5NWTU1Py5d9U'"

    url =
    f"https://API.telegram.org/bot{TOKEN}/sendMessage?chat_id={chat_
    id}&text={text}"

    channel_id = "-1001526803460"

    url += token

    method = url + "/sendMessage"

    requests.get
    'https://API.telegram.org/bot{}/
    sendMessage'.format(token),
    params=dict('channel_id'))

    r = requests.post(method,
        data={
            "chat_id": channel_id,
            "name": name,
            "text": text})

    if r.status_code != 200:
        raise Exception
        ("post_text error")
```

1.6 Описание работы программы

Набор действий предоставленных пользователю при взаимодействии с программой изображен на рисунке 1.2. Был разработан интерфейс приложения, который будет интуитивно понятен каждому пользователю. В главном меню представлены три основные кнопки (рисунок 1.3), которые и отвечают за три основных модуля программы:

					ПГУ 1.090301.28.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

- обнаружение и удаление;
- обратная связь;
- сканирование файла.

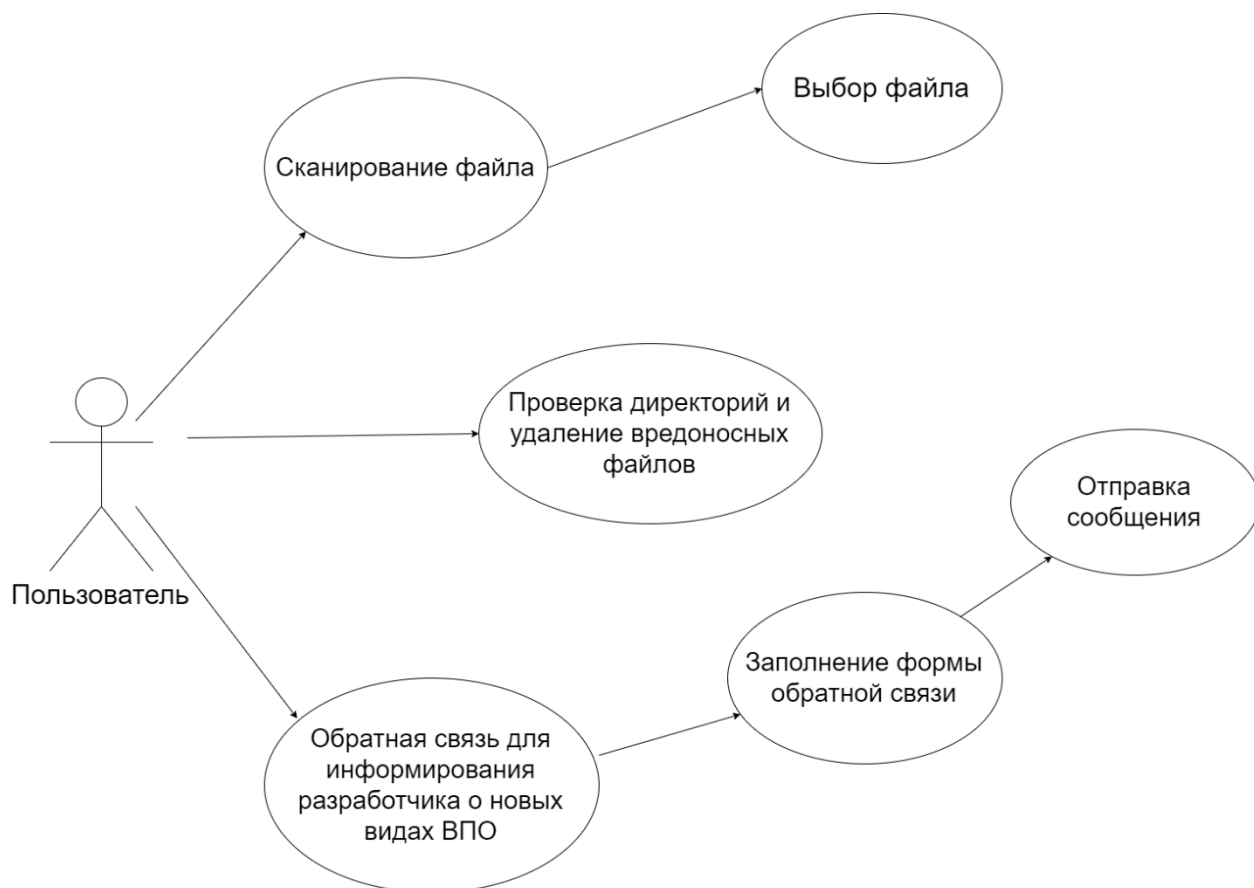


Рисунок 1.2 – UML диаграмма взаимодействия

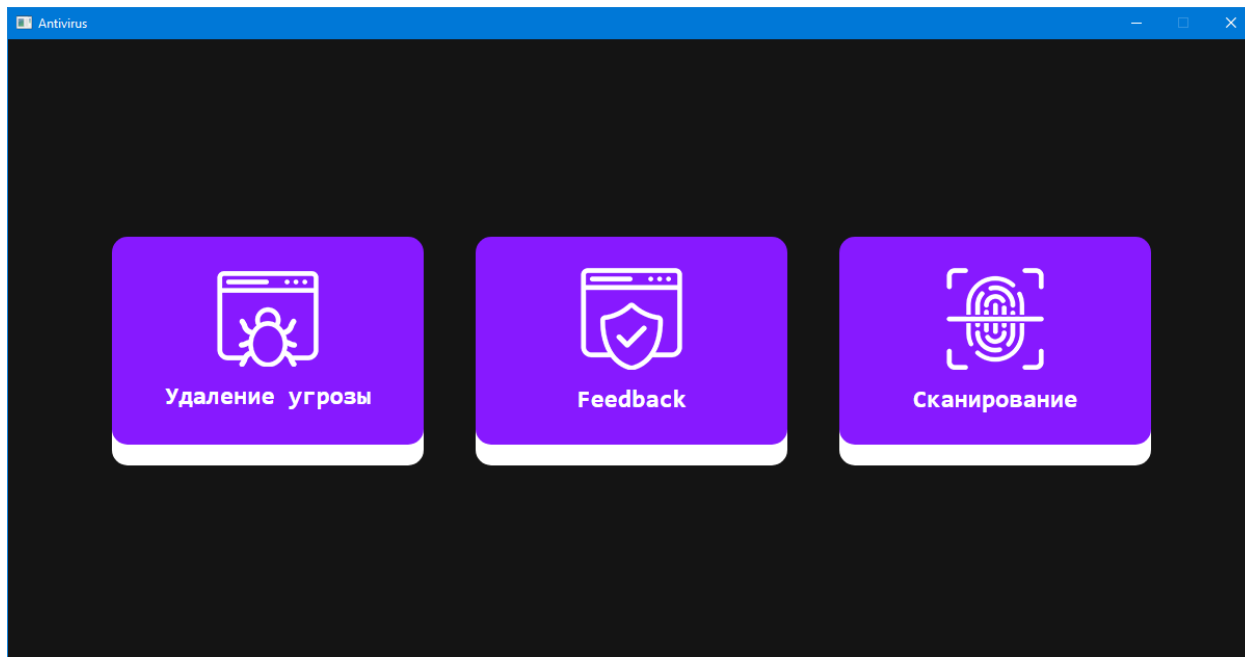


Рисунок 1.3 – Главное меню программы

При наведении на каждую из кнопок раскрывается карточка, в которой есть краткое описание данных кнопок и конкретных функций которые они выполняют (рисунок 1.4-1.6).

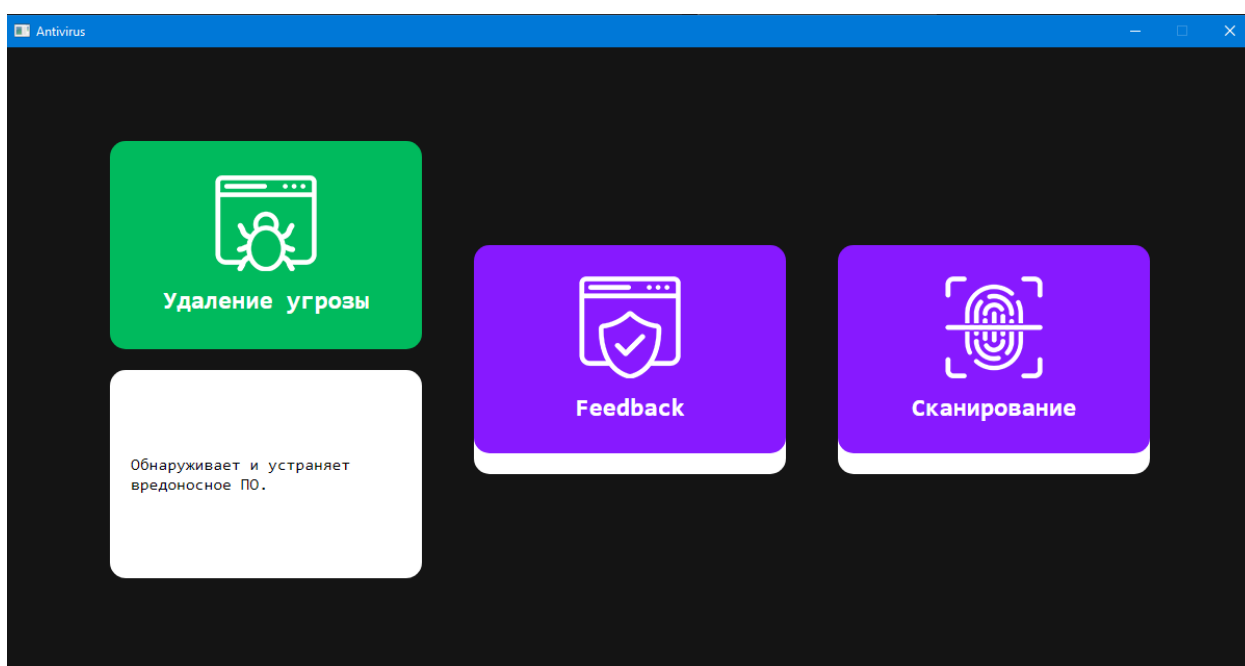


Рисунок 1.4 – Кнопка «Удаление угрозы»

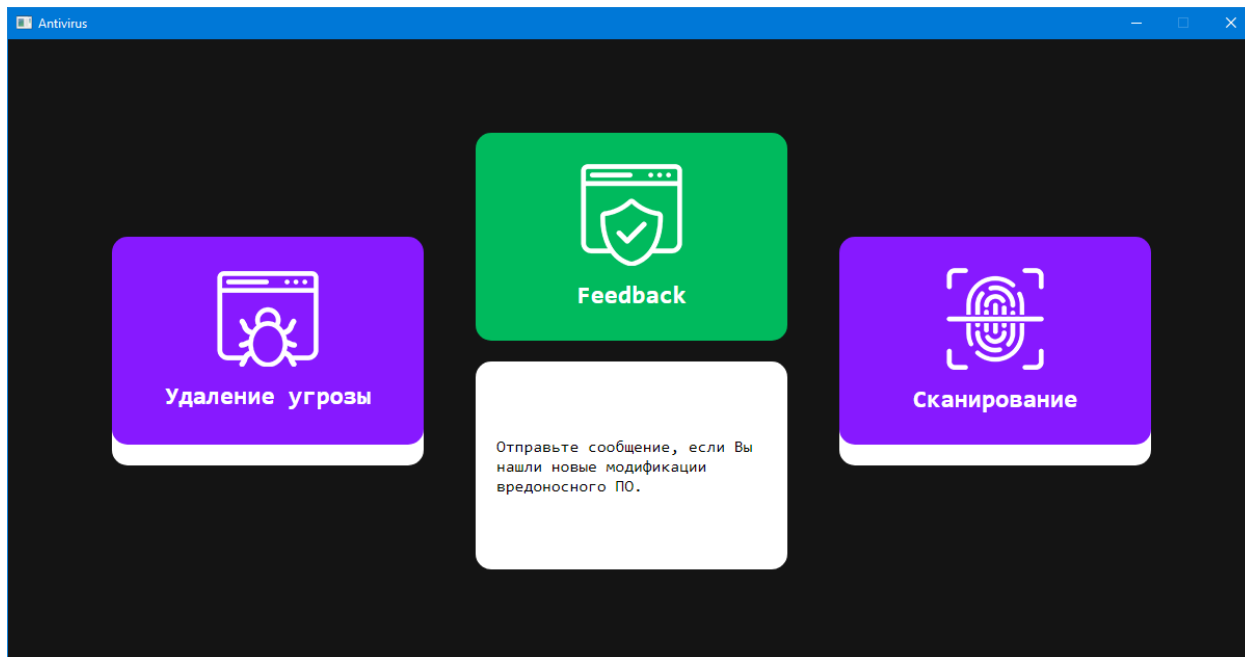


Рисунок 1.5 – Кнопка «Feedback»

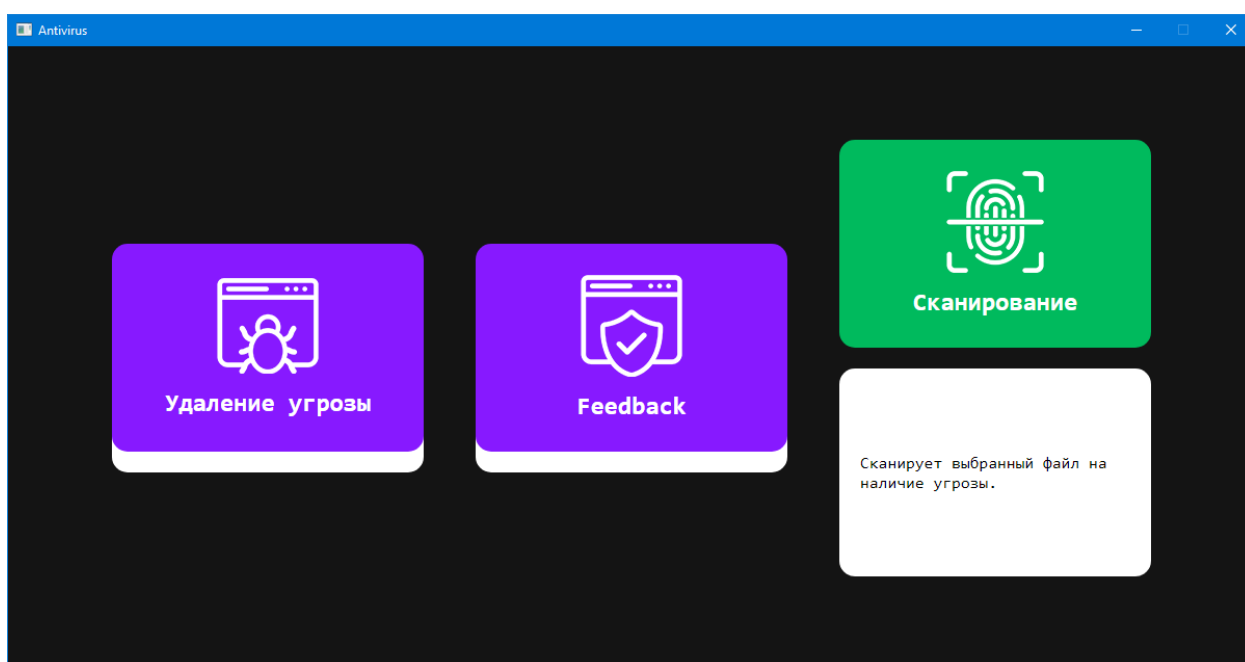


Рисунок 1.6 – Кнопка «Сканирование»

При нажатии кнопки «Удаление угрозы» (рисунок 1.7), запускается функция *Clear*, которая начинает проводить проверку внутри определенных директорий, которые могут быть «заражены». Пользователь видит прогресс обнаружения и уведомление об удалении угрозы (рисунок 1.8).

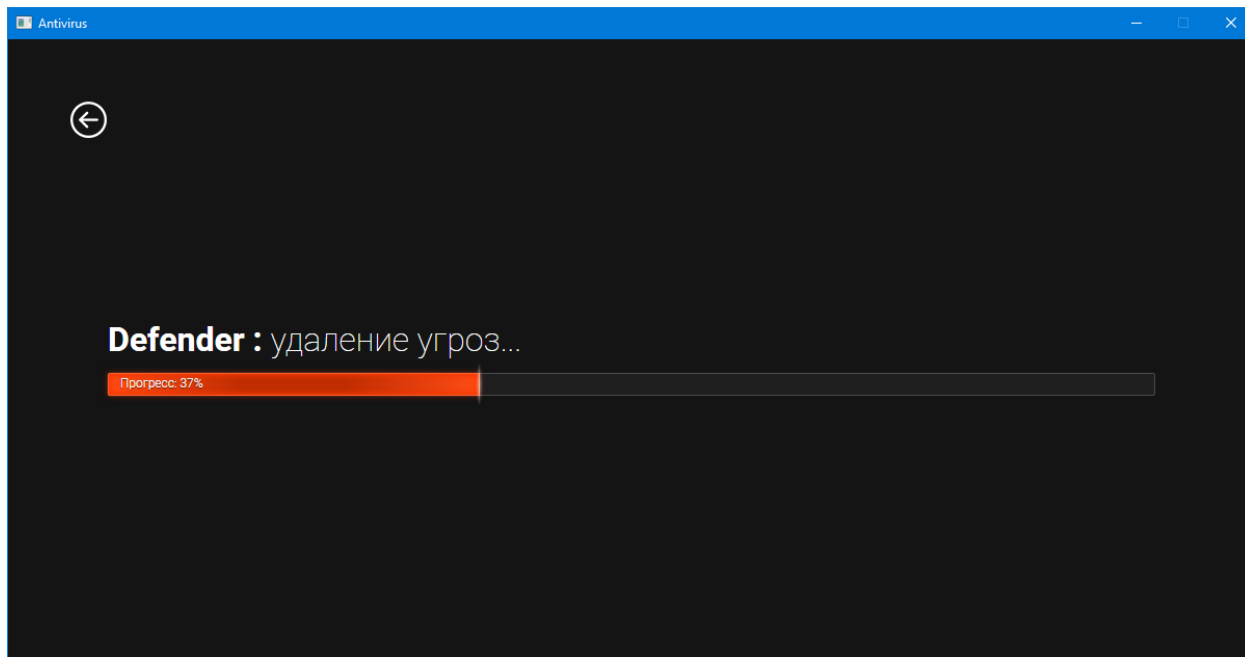


Рисунок 1.7 – Прогресс при запуске «Удаление угроз»

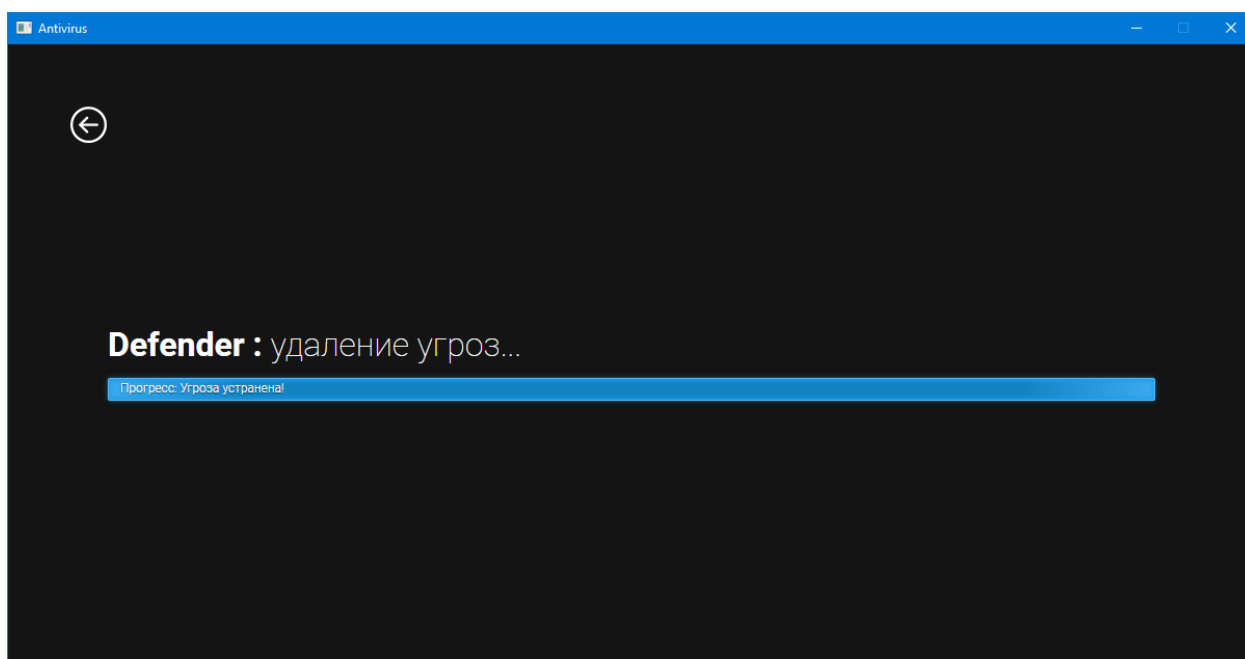


Рисунок 1.8 – Уведомление об устраненной угрозе

При нажатии на кнопку «*Feedback*» (рисунок 1.9), открывается форма обратной связи. Пользователь вводит своё имя или же для удобства *username* в *Telegram*, а также пишет своё сообщение по обнаруженным новым модификациям вируса или же недочёты самой программы.

					ПГУ 1.090301.28.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

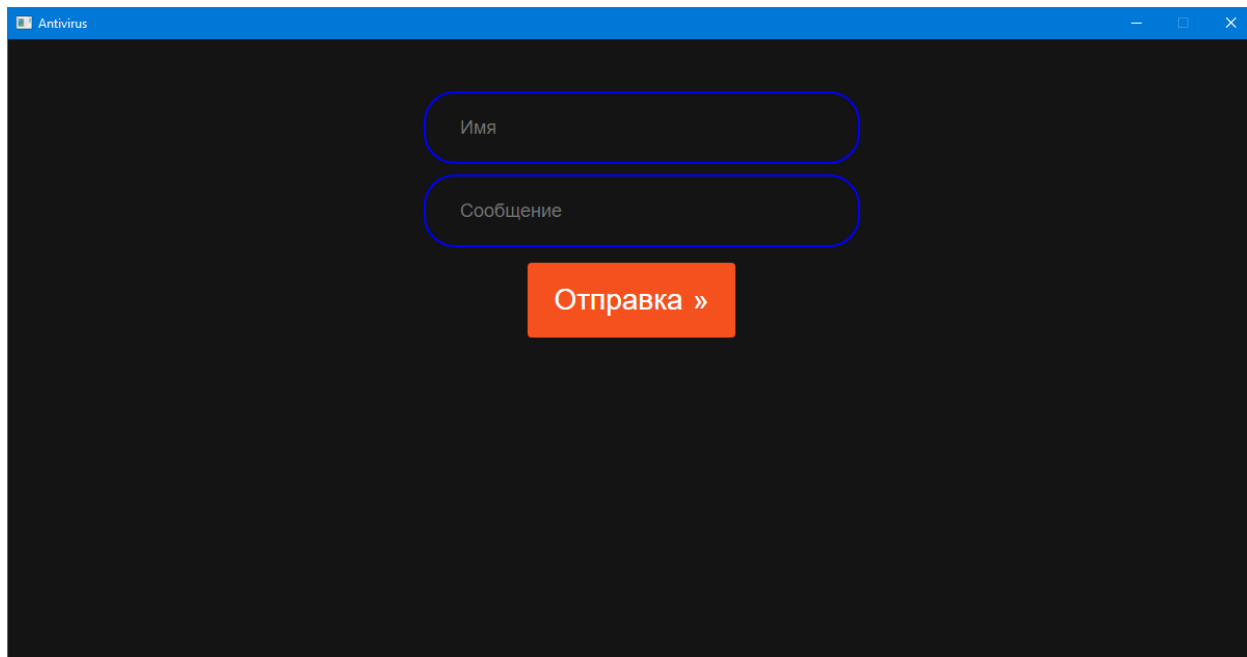


Рисунок 1.9 – Форма обратной связи

Сообщения доставляются непосредственно в канал разработчика программы (рисунок 1.10).

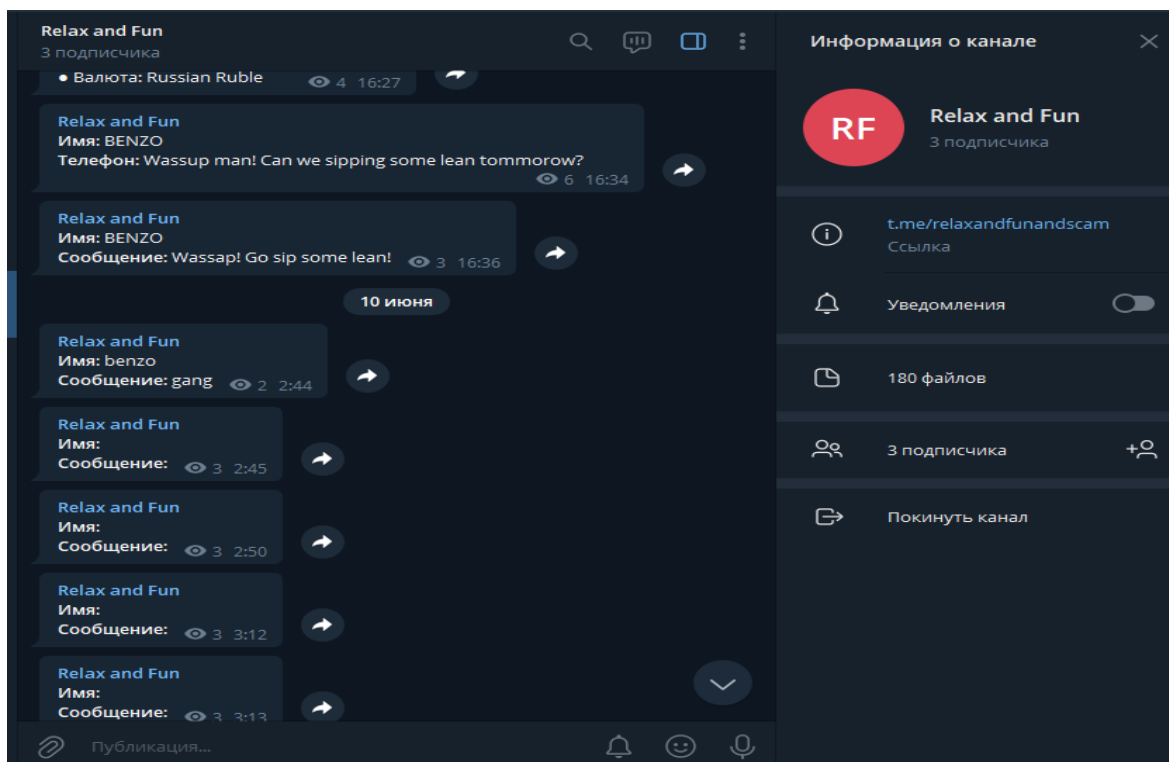


Рисунок 1.10 – Отображение сообщений в канале *Telegram*

При нажатии на кнопку «Сканирование» (рисунок 1.11), открывается окно загрузки проверяемого файла.

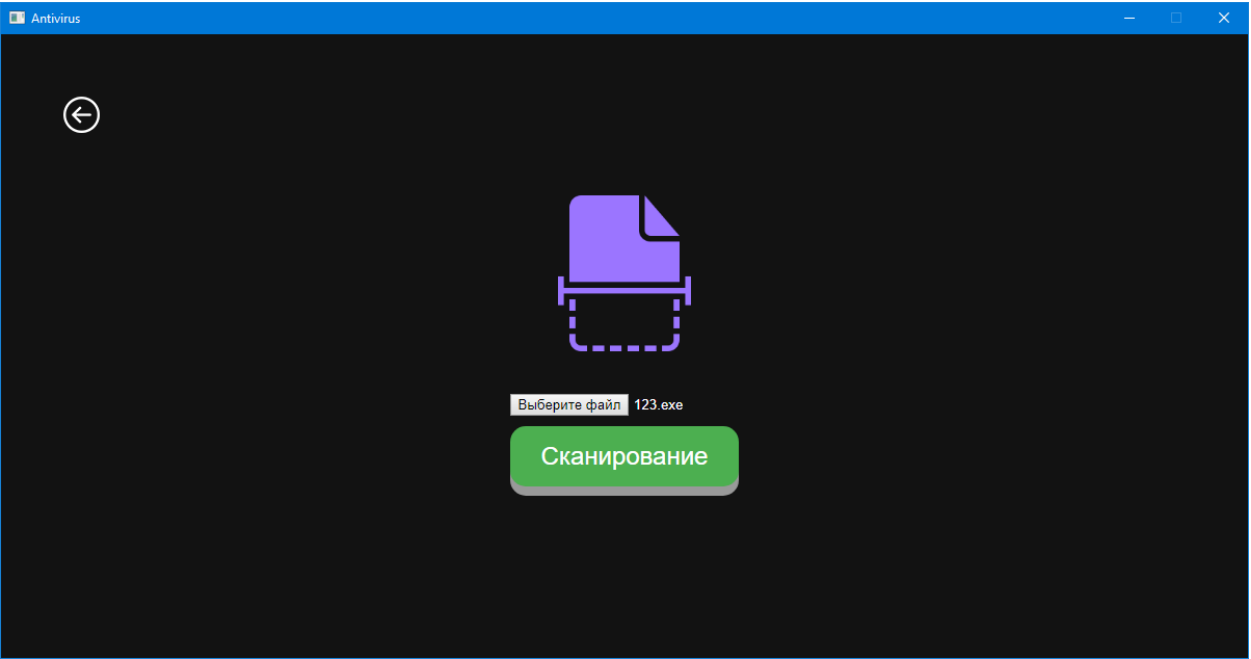


Рисунок 1.11 – Окно загрузки файла для сканирования

После выбора файла и нажатия на кнопку «Сканировать», открывается окно обнаружения угрозы и реализуется функция *Scan*. Пользователь видит прогресс обнаружения угрозы в проверяемом файле. (рисунок 1.12)

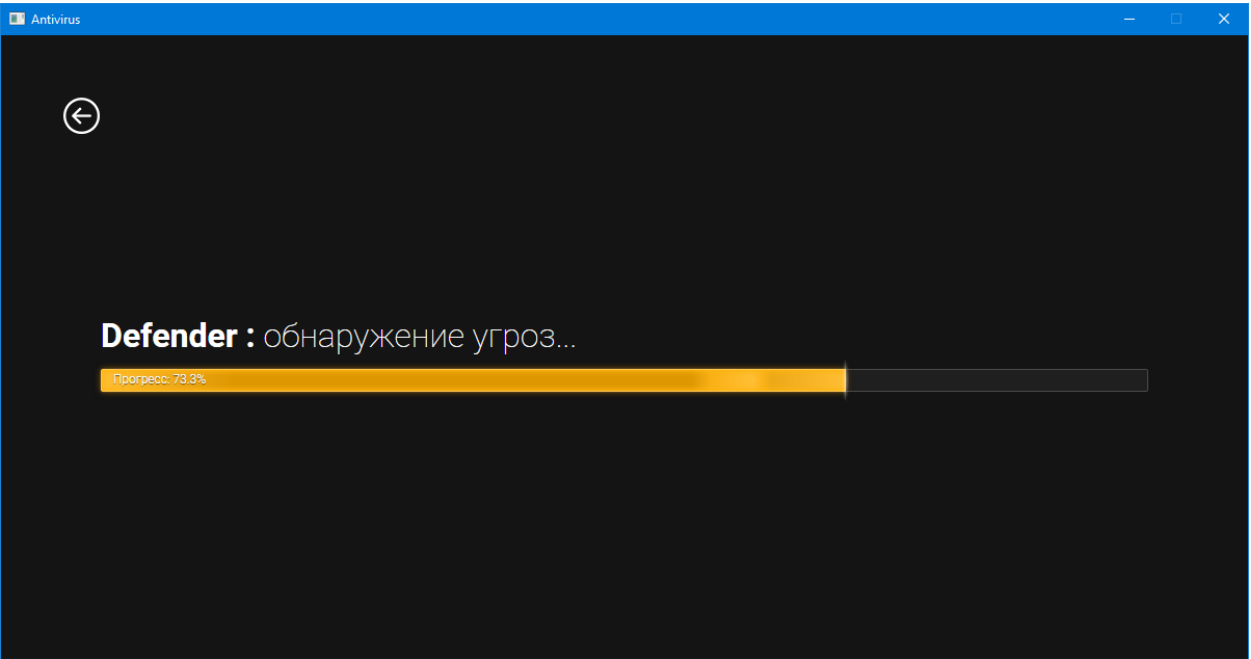


Рисунок 1.12 – Прогресс при сканировании файла

При обнаруженной угрозе в файле, пользователь видит «Угроза обнаружена!» в строке состояния (рисунок 1.13).

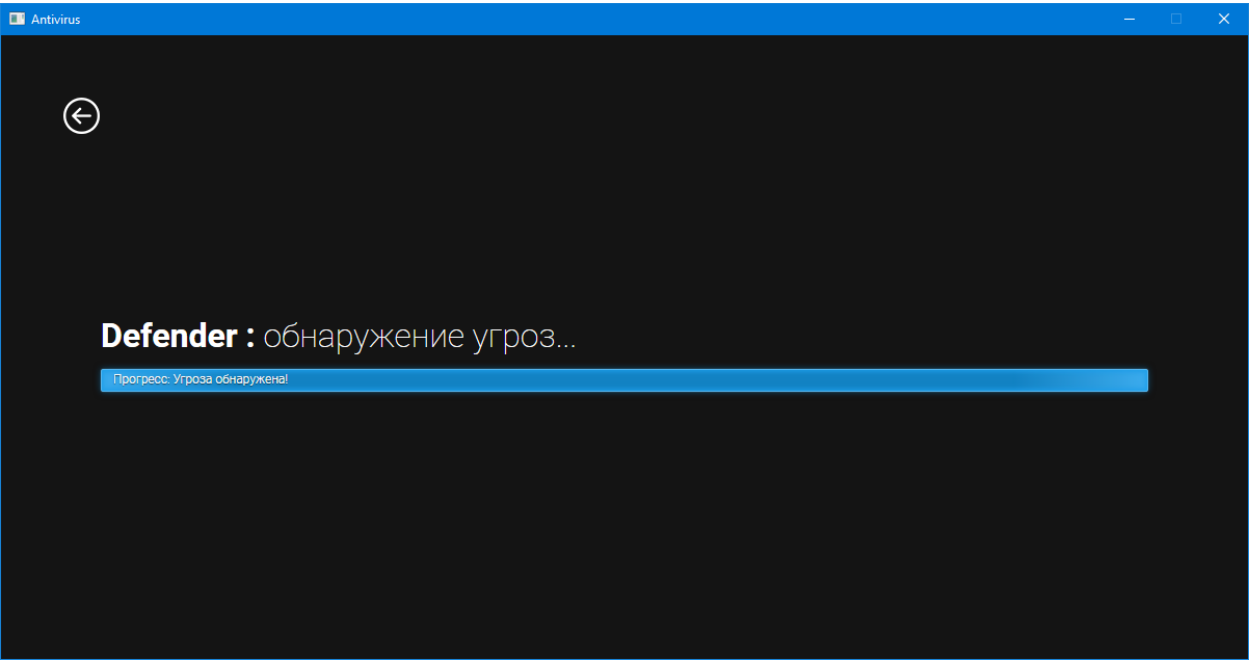


Рисунок 1.13 – Уведомление об обнаруженной угрозе при сканировании

2 Охрана труда

2.1 Анализ негативных факторов, воздействующих на человека

Чрезмерное использование компьютера и интернета может привести к нескольким проблемам со здоровьем. Расширенное использование компьютера может влиять как на физиологические, так и на психические факторы здоровья. Это сопровождается плохой осанкой, скелетно-мышечными болями, ожирением, слабым зрением и психическими заболеваниями. Опасность для здоровья использования компьютера жизнь требует серьезного беспокойства.

Для уменьшения пагубного воздействия ПЭВМ на человека, необходимо соблюдать ряд правил использования персонального компьютера. Главными являются:

- правильная эргономика рабочего места;
- правила перерывов и режима работы;
- правила электробезопасности;
- освещенность рабочего места.

2.1.1 Эргономика рабочего места

Ниже представлены основные требования, которым должно удовлетворять рабочее место:

- необходим стул, который поддерживает изгибы позвоночника, где можно отрегулировать высоту так, чтобы ноги лежали на полу или на подставке для ног, а бедра были параллельны полу. Подлокотники должны находится в том положение, в котором руки мягко опираются на них, а плечи расслаблены;

- местоположение мыши должно находиться в легкой досягаемости и на той же поверхности, что и клавиатура. При наборе текста или использовании

мышцы запястья нужно держать прямыми, предплечье близко к телу, а руки на уровне локтей или чуть ниже;

- под столом должен быть зазор для коленей, бедер и ног. Если стол слишком низкий и его нельзя отрегулировать. Если стол слишком высокий и его нельзя отрегулировать, то нужно поднять стул;

- монитор необходимо расположить на расстоянии вытянутой руки. Верхняя часть экрана должна быть на уровне глаз или чуть ниже. Монитор должен находиться непосредственно за клавиатурой. При ношении бифокальных очков, нужно опустить монитор еще на 1-2 дюйма для более удобного просмотра. Монитор располагается так, чтобы самый яркий источник света находился сбоку.

2.1.2 Правила перерывов и режима работы

Работник, работающий с ПК, обязан делать перерывы, чтобы быть более эффективным и не переутомляться. Для этого он должен:

- сократить длительность компьютерного времени, когда это возможно;
- разбить работу на более мелкие сегменты и переключаться между задачами, которые используют разные движения. Например, чередовать использование мыши с чтением и поиском в Интернете;

- по крайней мере каждые 10 минут делать короткий (10-20 секунд) перерыв. Убирать руки с клавиатуры и двигаться;

- каждые 30-60 минут делать короткий (2-5 минут) перерыв, чтобы потянуться или прогуляться.

2.1.3 Электробезопасность

ПЭВМ подключена к электросети с напряжением 220 В, поэтому представляет для человека потенциальную опасность. Любое воздействие тока может привести к сильному ожогу и к нарушению работы внутренних органов.

Для предотвращения подобных ситуаций необходимо соблюдать требования безопасности, прописанные в ГОСТ 12.2.007-75. Основные меры предосторожности для соблюдения:

					ПГУ 1.090301.28.001 ПЗ	Лист
						32
Изм.	Лист	№ докум.	Подпись	Дата		

- розетки электропитания должны соответствовать вилкам электропитания;

- во время работы персонального компьютера запрещается подключать и отключать кабели соединения;

- все электрические розетки должны иметь маркировку по напряжению;

- значение номинального напряжения сети необходимо наносить яркой краской, крупными символами (высотой не менее 50 мм) на стене или щите, возле розетки.

Во время компьютерной техники каждый сотрудник должен внимательно и ответственно относиться к электропроводке, приборам и аппаратам, всегда помнить, что несоблюдение правил электробезопасности угрожает здоровью и жизни человека.

Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть.

При обнаружении неисправностей необходимо обесточить электрооборудование. Продолжение работы возможно только после устранения всех неисправностей.

2.1.4 Освещенность рабочего места

Как естественное, так и искусственное (местное) освещение влияет на наше здоровье и благополучие. Это напрямую влияет на наше настроение и бдительность. Влияние освещения на зрение является наиболее очевидным воздействием на рабочую силу. Проблемы со зрением и дискомфорт возникают, когда освещение затрудняет работникам видеть то, что они должны видеть при выполнении своей работы, что, в свою очередь, может негативно повлиять на их производительность. Аспекты освещения, которые обычно могут вызывать визуальный дискомфорт, - это недостаточное или чрезмерное освещение, тени, вуалирующие отражения, блики и мерцание. Исходя из этого, рассчитаем параметры искусственного освещения.

					ПГУ 1.090301.28.001 ПЗ	Лист
						33
Изм.	Лист	№ докум.	Подпись	Дата		

Расчет освещения производится для комнаты: длина (а) - 6 м, ширина (б) - 3 м, высота (h1) – 3 м, высота стола (h2) – 0.8 м. Расчет производится под типовой растровый светильник с четырьмя линейными люминесцентными лампами (Кл), мощностью 18 Вт каждая (1 лампа даёт световой поток (СПл), равный 900 лк).

Поскольку должность программиста предполагает длительные монотонные операции — с высоким уровнем зрительной работы (различение объектов, размером от 3 до 5 мм), то возьмём за норму – освещённость его рабочего места (Е) в 500 люкс.

Коэффициент запаса (Кз) обычно равен 1,2.

Определяем индекс помещения (Ип), расчет:

$$Ип = S/((h1-h2) * (a+b)) = (6*3)/((3-0,8)*(6+3)) \approx 0,9.$$

Для данного индекса помещения коэффициент использования (U) примерно равен 65.

Выполним, расчет количества светильников (Ксв.) для данного помещения:

$$Ксв. = (E*S*100*Kз)/(U*Kл*СПл).$$

Следовательно:

$$Ксв. = (500*18*100*1,2)/(65*4*900) = 5 \text{ светильников.}$$

Получается, что для кабинета программиста размером 6м х 3м достаточно будет установить 5 растровых светильников.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана система обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам.

На основании проделанной работы программа позволяет:

- обнаруживать вредоносное ПО несанкционированного доступа к криптокошелькам;
- устранять найденные угрозы;
- иметь удобный и интуитивно понятный пользовательский интерфейс;
- сканировать файлы на наличие «вируса»;
- отправлять информацию пользователем для информирования разработчика о возможных недоработках программы или о новых модификациях ВПО.

Во время выполнения данной работы были получены навыки разработки программы на языке *Python* с использованием библиотеки *PyQT*, а также реализации обратной связи с помощью *API* мессенджера *Telegram*.

Программа написана с использованием среды программирования *Sublime Text*.

В дальнейшем можно усовершенствовать разработанную программу и добавить новые функции, который уже защищали бы пользовательское устройство от всех видов вирусов. Это можно реализовать с помощью *API VirusTotal*, который является важнейшим сервисом при сканировании файлов, так как он показывает детекты всех известных антивирусов на данный момент.

Также возможна защита браузерных файлов *cookies* и настройка активного фаервола, который бы уведомлял пользователя о подозрительной сетевой активности.

Список использованных источников

1. Scammer.pro [Электронный ресурс]. URL: [https:// scammer.pro /](https://scammer.pro/)(дата обращения:27.04.2022).
2. Stackoverflow.com [Электронный ресурс]. URL: <https://ru.stackoverflow.com/> (дата обращения:23.04.2022).
3. Deepcloud.ru [Электронный ресурс]. URL: <https://deepcloud.ru/> (дата обращения:21.04.2022).
4. Wikipedia.org [Электронный ресурс]. URL: <https://ru.wikipedia.org/> (дата обращения:28.04.2022).
5. Python-scripts.com [Электронный ресурс]. URL: <https://python-scripts.com/> (дата обращения:30.04.2022).

Приложение А

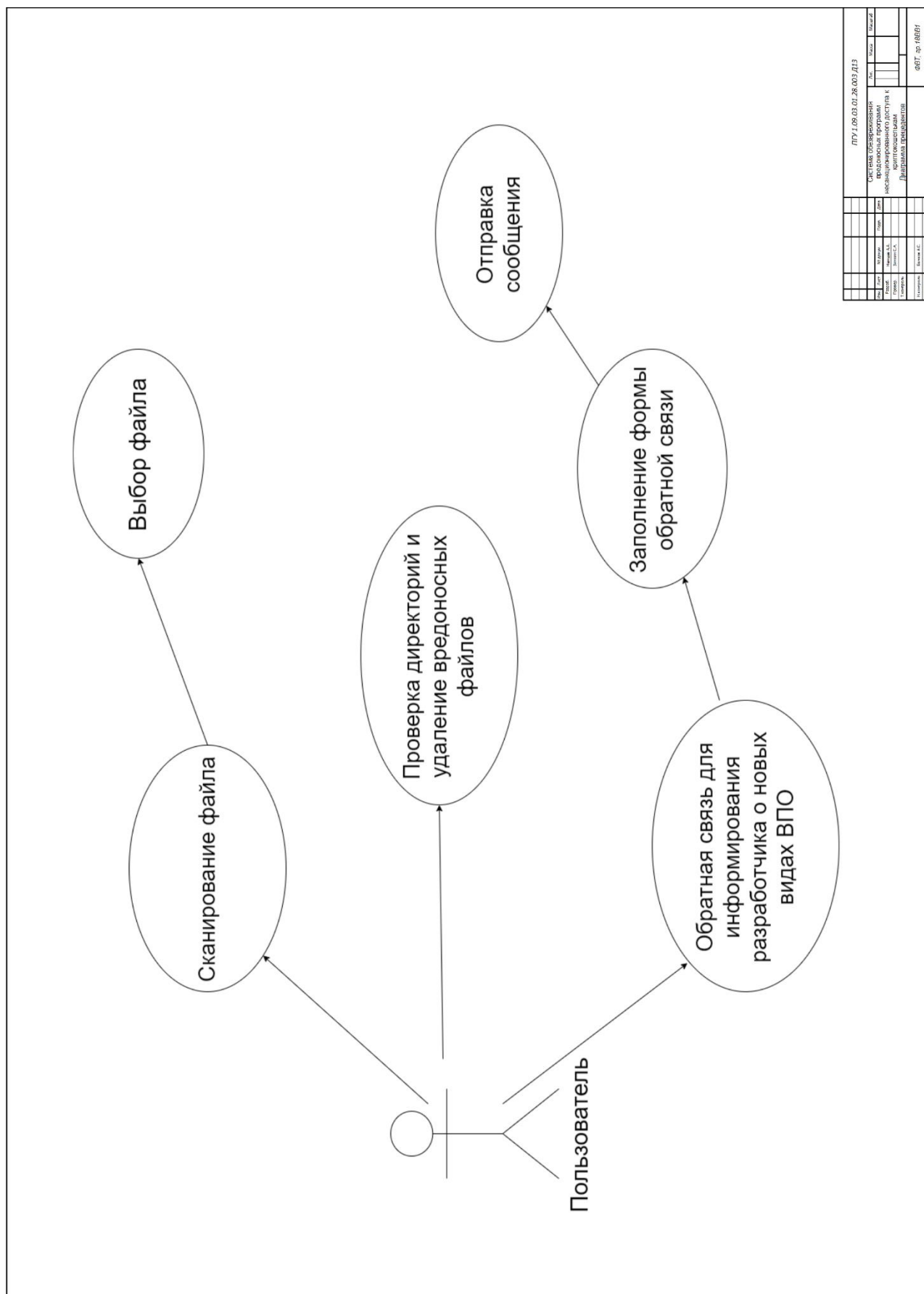
(Обязательное)

Система обезвреживания вредоносных программ несанкционированного
доступа к криптокошелькам

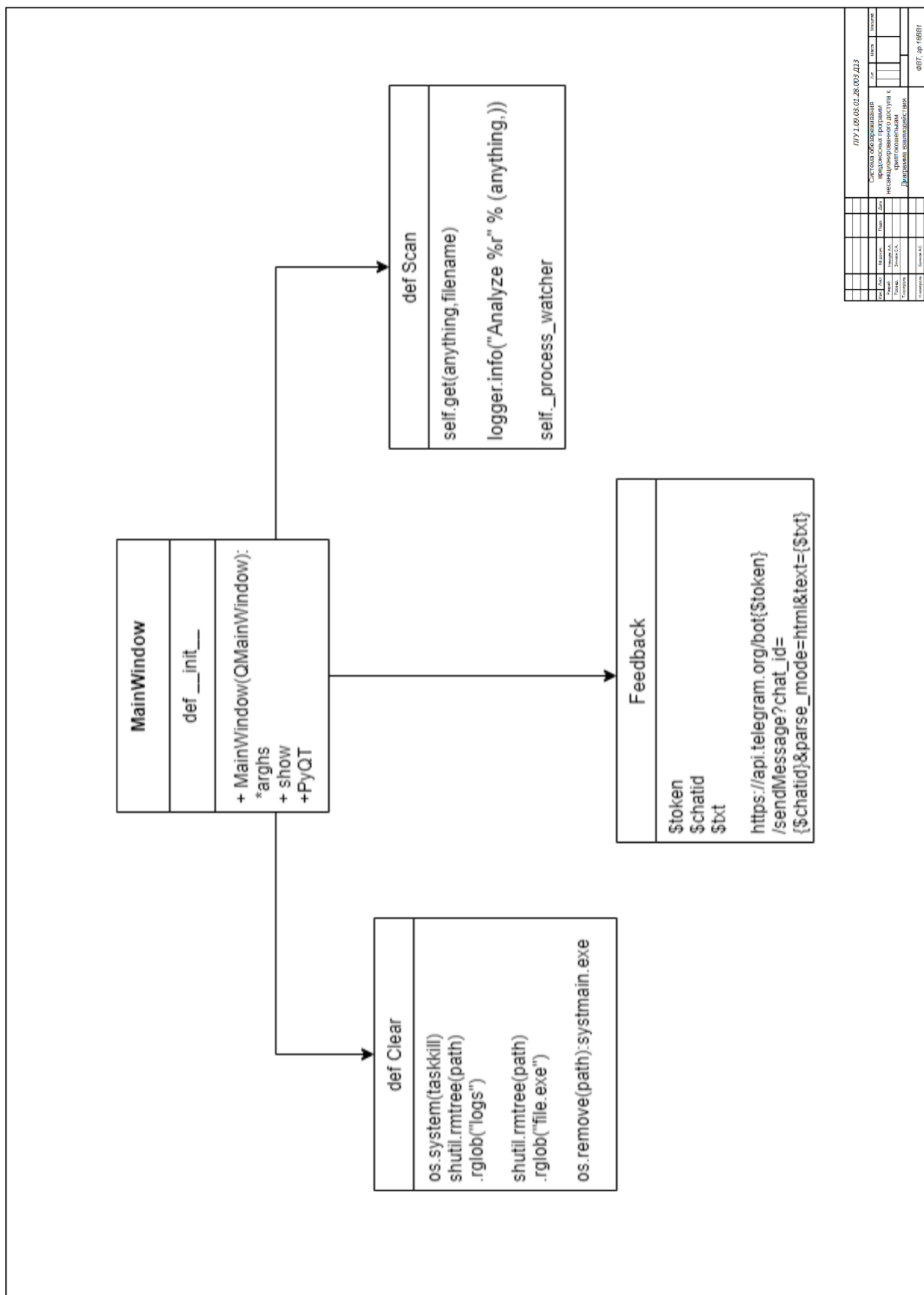
UML - диаграммы

					ПГУ 1.090301.28.001 ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

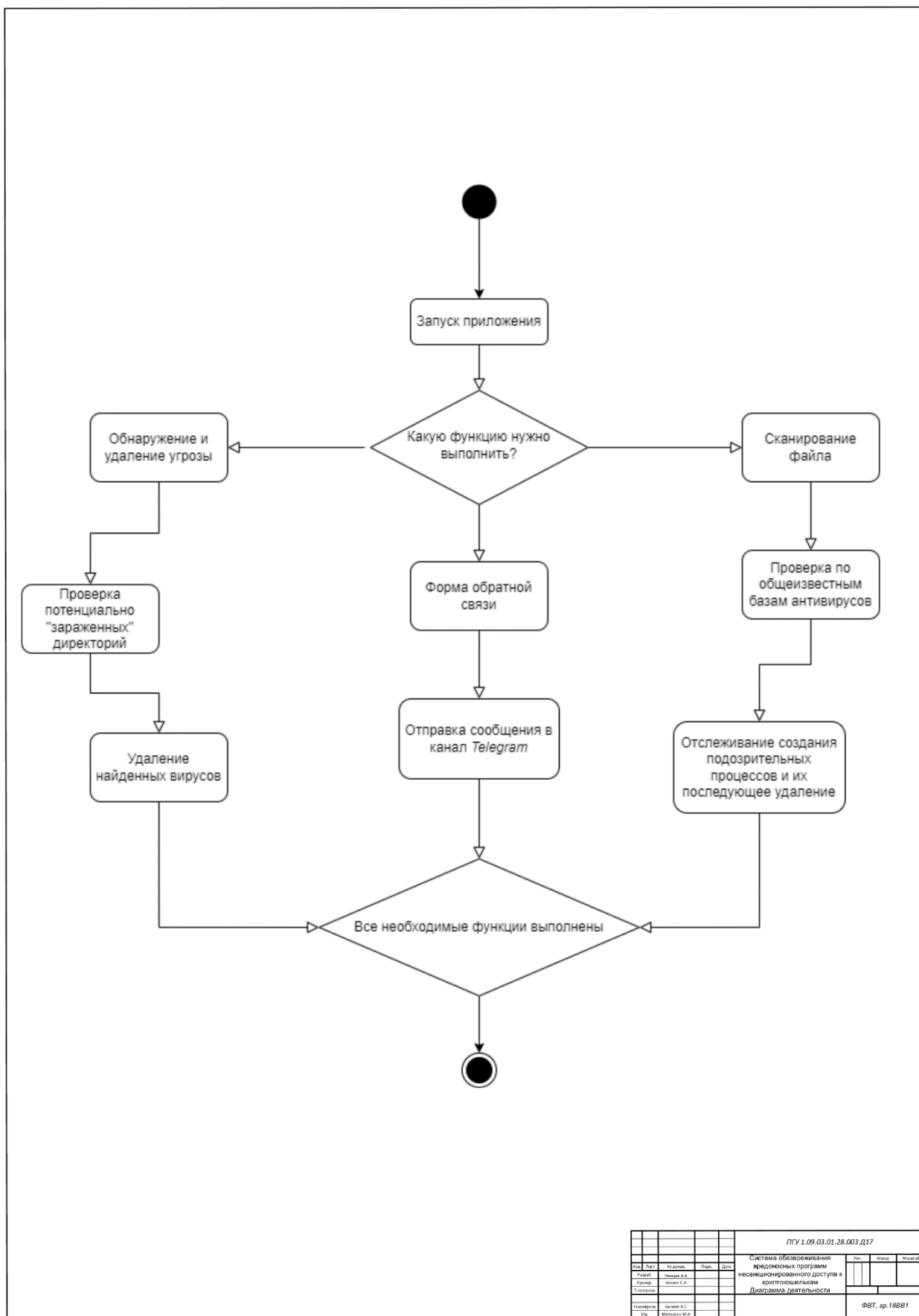
Приложение А.1 – Диаграмма прецедентов



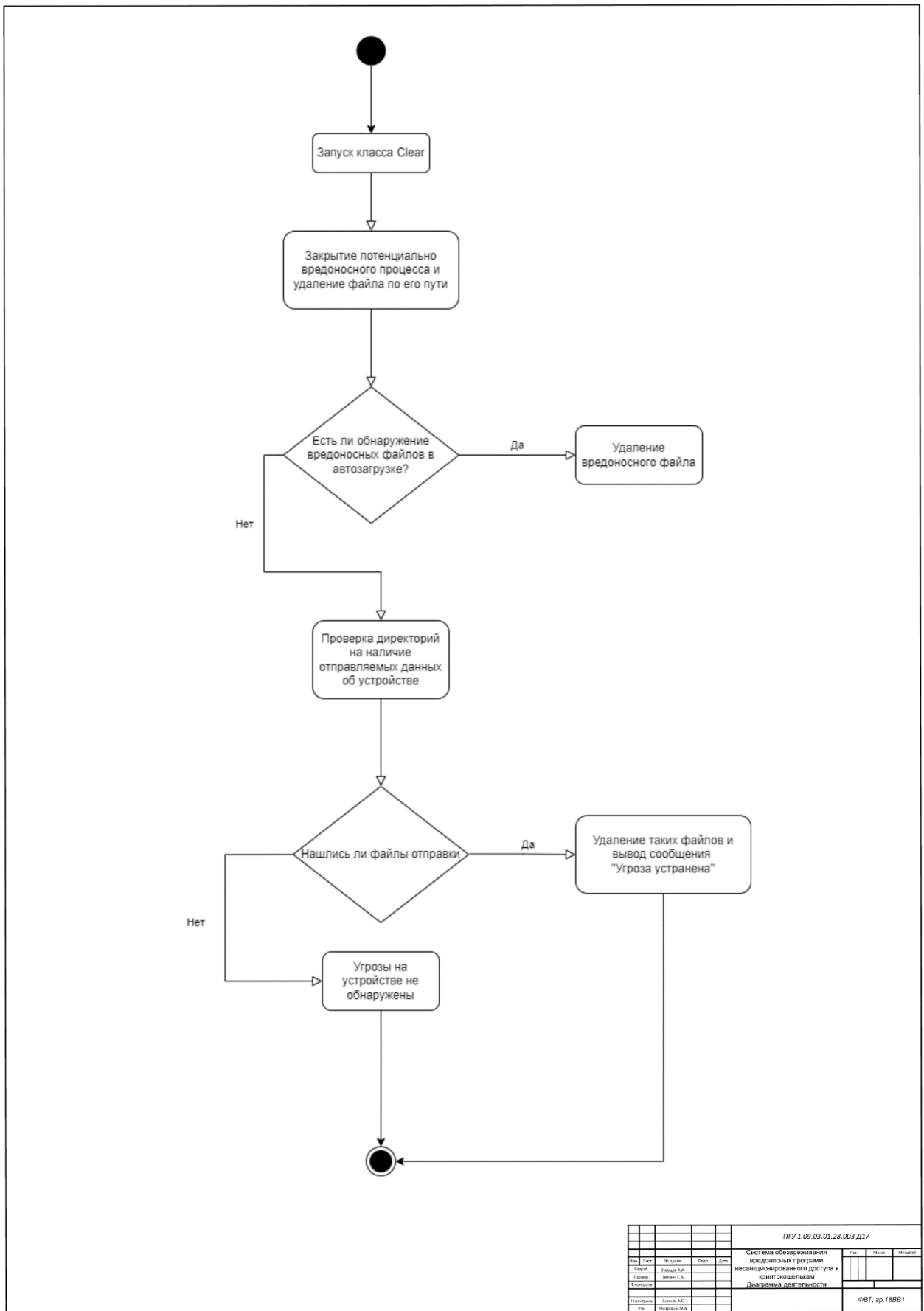
Приложение А.2 – Диаграмма классов



Приложение А.3 – Диаграмма деятельности



Приложение А.4 – Диаграмма деятельности модуля *Clear*



Приложение Б

(обязательное)

Система обезвреживания вредоносных программ несанкционированного
доступа к криптокошелькам

Листинг программы

					ПГУ 1.090301.28.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «antivirus.py»

```

from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtWebEngineWidgets import QWebEngineView
from PyQt5.QtCore import QUrl
import sys
import psutil
import shutil
import time
import os
from threading import Thread
from pathlib import Path
import getpass

login = getpass.getuser()

def f_1():
    class MainWindow(QMainWindow):
        def __init__(self, *args, **kwargs):
            super(MainWindow, self).__init__(*args, **kwargs)
            self.setFixedSize(1000, 200)
            self.setWindowTitle('Antivirus')
            self.show()

    app = QApplication(sys.argv)
    app.setStyle('Fusion')
    window = MainWindow()
    app.exec_()

def f_2():
    time.sleep(10)
    os.system("taskkill /im 123.exe")
    try:
        shutil.rmtree((next(Path('C:/ProgramData/MicrosoftStorage/').rglob("logs"))))
    except (StopIteration, PermissionError): pass
    try:
        shutil.rmtree((next(Path(r'C://Users/%s/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/' % login).rglob("123.exe"))))
    except (StopIteration, PermissionError): pass
    try: os.remove(r"C:/Users/%s/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/systmain.exe" % login)
    except FileNotFoundError: pass

```

					ПГУ 1.090301.28.001 ПЗ	Лист
						43
Изм.	Лист	№ докум.	Подпись	Дата		

```

    os.system('reg delete
"HKEY_USER_MACHINE/SOFTWARE/Microsoft/Windows/CurrentVersion/Run/virus.exe")

    shutil.copy(r'123.exe',
r'C://Users/%s/AppData/Roaming/Microsoft/Windows/Start
Menu/Programs/Startup/systmain.exe' % login)

    os.chdir('C:/Users/'+login+'/AppData/Roaming/Microsoft/Windows/Start
Menu/Programs/Startup')

class FileMonitor:

    def __init__(self, notify_filter, **kwargs):
        valid_kwargs = ('Drive', 'Path', 'FileName', 'Extension')
        for key in kwargs:
            if key not in valid_kwargs:
                raise FileMonitorError(
                    )

        self._notify_filter = notify_filter
        self._kwargs = kwargs
        self._event_properties = {
            'Drive': None,
            'Path': None,
            'FileName': None,
            'Extension': None,
            'Timestamp': None,
            'EventType': None,
        }

    @property
    def drive(self):
        return self._event_properties['Drive']

    @property
    def path(self):
        return self._event_properties['Path']

    @property
    def file_name(self):
        return self._event_properties['FileName']

    @property
    def extension(self):
        return self._event_properties['Extension']

    @property

```

```

def timestamp(self):
    return self._event_properties['Timestamp']
@property
def event_type(self):
    return self._event_properties['EventType']
class FileMonitorAPI(FileMonitor):
    """Class defines a files monitor object."""
    _ACTIONS = {
        0x00000000: 'Unknown action',
        0x00000001: 'Added',
        0x00000002: 'Removed',
        0x00000003: 'Modified',
        0x00000004: 'Renamed from file or directory',
        0x00000005: 'Renamed to file or directory'
    }
    FileMonitor.__init__(self, notify_filter, **kwargs)
    valid_notify_filter = (
        'FileNameChange',
        'DirNameChange',
        'LastWriteChange',
        'UnionChange'
    )
    if self._notify_filter not in valid_notify_filter:
        raise FileMonitorError(
            'Watcher installation error.'
            'The notify_filter value cannot be: "{}".'.
            format(self._notify_filter)
        )
    try:
        self._directory = win32file.CreateFile(
            self._kwargs['Path'],
            winnt.FILE_LIST_DIRECTORY,
            win32con.FILE_SHARE_READ |
            win32con.FILE_SHARE_WRITE,
            None,
            win32con.OPEN_EXISTING,
            win32con.FILE_FLAG_BACKUP_SEMANTICS |

```

```

        win32con.FILE_FLAG_OVERLAPPED,
        None
    )
except pywintypes.error as err:
    raise FileMonitorError(
        'Failed to open directory. Error code: {}'.format(err.winerror)
    ) from err
self._overlapped = pywintypes.OVERLAPPED()
self._overlapped.hEvent = win32event.CreateEvent(
    None,
    False,
    False,
    None
)
if not self._overlapped.hEvent:
    raise FileMonitorError(
        'Failed to create event. Error code: {}'.format(self._overlapped.hEvent)
    )
self._buffer = win32file.AllocateReadBuffer(1024)
self._num_bytes_returned = 0
self._set_watcher()

def update(self):
    while True:
        result = win32event.WaitForSingleObject(self._overlapped.hEvent,
0)

        if result == win32con.WAIT_OBJECT_0:
            self._num_bytes_returned = win32file.GetOverlappedResult(
                self._directory,
                self._overlapped,
                True
            )
            timestamp = datetime.datetime.fromtimestamp(
                datetime.datetime.utcnow().timestamp()
            )

```

```

        self._event_properties['Path'] = self._get_path()
        self._event_properties['FileName'] = self._get_file_name()
        self._event_properties['Timestamp'] = timestamp
        self._event_properties['EventType'] = self._get_event_type()
        self._set_watcher()
        break

    if result == win32con.WAIT_FAILED:
        self.close()
        raise FileMonitorError()

def close(self):
    """Close the open directory and event handles."""
    win32api.CloseHandle(self._kwargs['Path'])
    win32api.CloseHandle(self._overlapped.hEvent)

def _get_notify_filter_const(self):
    if self._notify_filter == 'FileNameChange':
        return win32con.FILE_NOTIFY_CHANGE_FILE_NAME
    if self._notify_filter == 'DirNameChange':
        return win32con.FILE_NOTIFY_CHANGE_DIR_NAME
    if self._notify_filter == 'LastWriteChange':
        return win32con.FILE_NOTIFY_CHANGE_LAST_WRITE
    if self._notify_filter == 'UnionChange':
        return (
            win32con.FILE_NOTIFY_CHANGE_FILE_NAME |
            win32con.FILE_NOTIFY_CHANGE_DIR_NAME |
            win32con.FILE_NOTIFY_CHANGE_LAST_WRITE
        )
    return None

def _get_event_type(self):
    result = ''
    if self._num_bytes_returned != 0:
        result = self._ACTIONS.get(win32file.FILE_NOTIFY_INFORMATION(
            self._buffer, self._num_bytes_returned)[0][0], 'Unknown')
    return result

```

```

def _get_path(self):
    result = ''
    if self._num_bytes_returned != 0:
        result = win32file.GetFinalPathNameByHandle(
            self._directory,
            win32con.FILE_NAME_NORMALIZED
        )
    return result

def _get_file_name(self):
    result = ''
    if self._num_bytes_returned != 0:
        result = win32file.FILE_NOTIFY_INFORMATION(
            self._buffer, self._num_bytes_returned)[0][1]
    return result

def _set_watcher(self):
    win32file.ReadDirectoryChangesW(
        self._directory,
        self._buffer,
        True,
        self._get_notify_filter_const(),
        self._overlapped,
        None
    )

class FileMonitorWMI(FileMonitor):

    def __init__(self, notify_obj='File', notify_filter='Operation',
        **kwargs):
        Args:
        FileMonitor.__init__(self, notify_filter, **kwargs)
        valid_notify_filter = (
            'Operation',
            'Creation',
            'Deletion',
            'Modification'

```

```

)

if self._notify_filter not in valid_notify_filter:
    raise FileMonitorError(
        'Watcher installation error.'
        'The notify_filter value cannot be: "{}".'.
        format(self._notify_filter)
    )

valid_notify_obj = ('File', 'Directory')
if notify_obj not in valid_notify_obj:
    raise FileMonitorError(
        'Watcher installation error.'
        'The notify_obj value cannot be: "{}".'.
        format(notify_obj)
    )

wmi_obj = wmi.WMI(namespace='root/CIMv2')
if notify_obj == 'File':
    try:
        self._watcher = wmi_obj.CIM_DataFile.watch_for(
            self._notify_filter,
            **kwargs
        )
    except wmi.x_wmi as err:
        raise FileMonitorError(
            'Watcher installation error. Error code: {}'.'.
            format(err.com_error.hresult or 'unknown')
        ) from err
elif notify_obj == 'Directory':
    try:
        self._watcher = wmi_obj.CIM_Directory.watch_for(
            self._notify_filter,
            **kwargs
        )
    except wmi.x_wmi as err:
        raise FileMonitorError(
            format(err.com_error.hresult or 'unknown')
        ) from err
else:

```

```

        raise FileMonitorError('Ошибка.')

def update(self):
    event = self._watcher()
    self._event_properties['Timestamp'] = event.timestamp
    self._event_properties['EventType'] = event.event_type
    if attributes(event, 'Drive'):
        self._event_properties['Drive'] = event.Drive
    if attributes(event, 'Path'):
        self._event_properties['Path'] = event.Path
    if attributes(event, 'FileName'):
        self._event_properties['FileName'] = event.FileName
    if attributes(event, 'Extension'):
        self._event_properties['Extension'] = event.Extension

class FileMonitorError(Exception):
    pass

def Clear():
    time.sleep(10)
    os.system("taskkill /im 123.exe")
    try:
        shutil.rmtree((next(Path('C:/ProgramData/MicrosoftStorage/').rglob("logs"))))
    except (StopIteration, PermissionError): pass
    try:
        shutil.rmtree((next(Path('C:/ProgramData/Disc-Soft/').rglob("logs"))))
    except (StopIteration, PermissionError): pass
    try:
        shutil.rmtree((next(Path('C:/ProgramData/MicrosoftDefender/').rglob("logs"))))
    except (StopIteration, PermissionError): pass
    try:
        shutil.rmtree((next(Path('C:/ProgramData/PackageCache/').rglob("logs"))))
    except (StopIteration, PermissionError): pass
    try:
        shutil.rmtree((next(Path('C:/ProgramData/Adobe/Installer/').rglob("logs"))))
    except (StopIteration, PermissionError): pass
    try:
        shutil.rmtree((next(Path(r'C://Users/%s/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/' % login).rglob("123.exe"))))
    except (StopIteration, PermissionError): pass

```

					ПГУ 1.090301.28.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

```

try: os.remove(r"C:/Users/%s/AppData/Roaming/Microsoft/Windows/Start
Menu/Programs/Startup/systmain.exe" %login)

except FileNotFoundError: pass

params = {'APIkey': self.API_key, 'resource': self.checksum}

reanalyze = reanalyze if reanalyze is not None else False

if not reanalyze:

    # Check if already exists

    report = self.get(anything, filename)

    if report is not None:

        return report

logger.info("Analyze %r" % (anything, ))

o = self._fileobj_to_fcontent(anything, filename)

assert o[0] == "file"

if o[1] is None:

    o[1] = "file"

if len(o[2]) > FILE_SIZE_LIMIT:

    raise self.EntityTooLarge()

self._limit_call_handler()

ret_json = postfile.post_multipart(

    host = "www.virustotal.com",

    selector = "https://www.virustotal.com/vtAPI/v2/file/scan",

    fields = {

        "APIkey": self.API_key,

    }.items(),

    files = [o,],

)

try:

    response = get(self.url, params=params)

    self.response = response.json()

except RequestException:

    exit(1)

except ValueError:

    exit(1)

if "scans" not in self.response:

    print_bar("Угроза не обнаружена", self.response["verbose_msg"],

5)

    exit(1)

```

```

        print_bar("Угроза обнаружена!",
                  "Итог: %progress " + str(self.response["positives"])
+ "/" + str(self.response["total"]) +
                  "\nFull report written to C:\\Users\\{}\\Scan report of
{}.txt".format(USERNAME, self.program_name), 10)

def feedback(text: str):
    token = "'5424533445:AAGzn0F7IyFKjRuFTV3BS5F5NWTU1Py5d9U'"
    url =
f"https://API.telegram.org/bot{TOKEN}/sendMessage?chat_id={chat_id}&text={tex
t}"
    channel_id = "-1001526803460"
    url += token
    method = url + "/sendMessage"
    requests.get
('https://API.telegram.org/bot{/}/sendMessage'.format(token),
params=dict('channel_id'))
    r = requests.post(method, data={
        "chat_id": channel_id,
        "name": name,
        "text": text})
    if r.status_code != 200:
        raise Exception("post_text error")
    th_1, th_2 = Thread(target=f_1), Thread(target = f_2)
    if __name__ == '__main__':
        th_1.start(), th_2.start()
        th_1.join(), th_2.join()

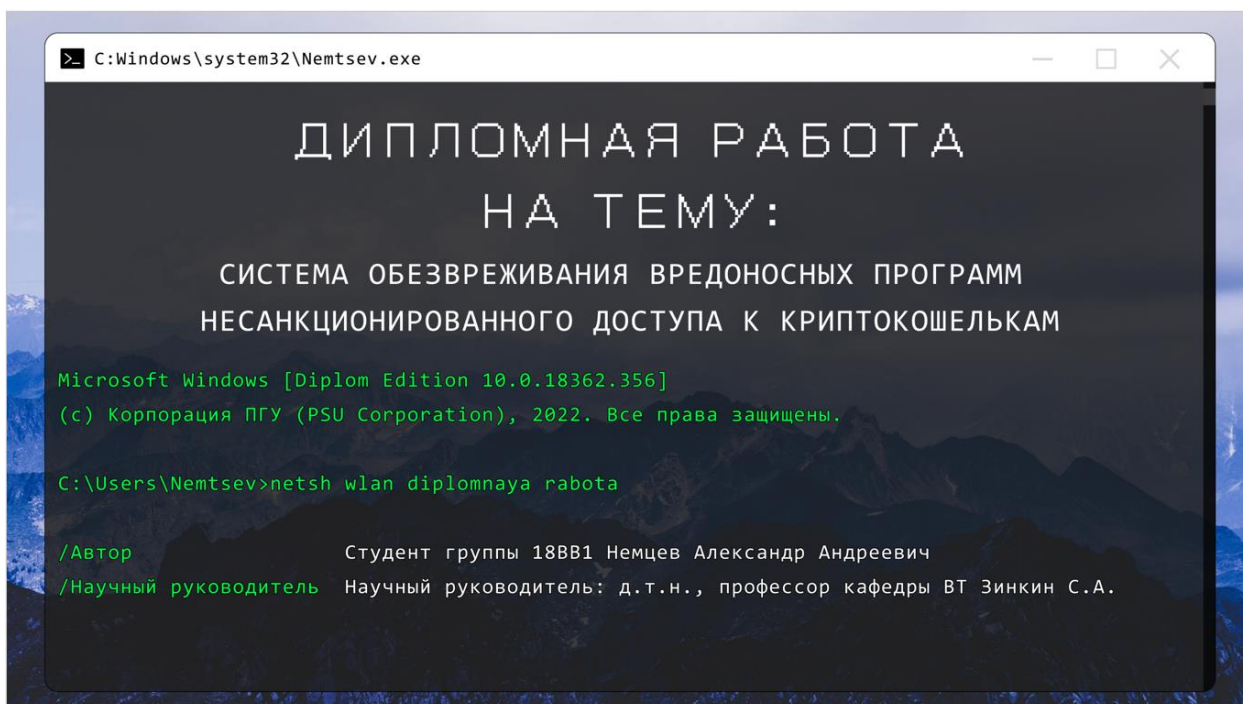
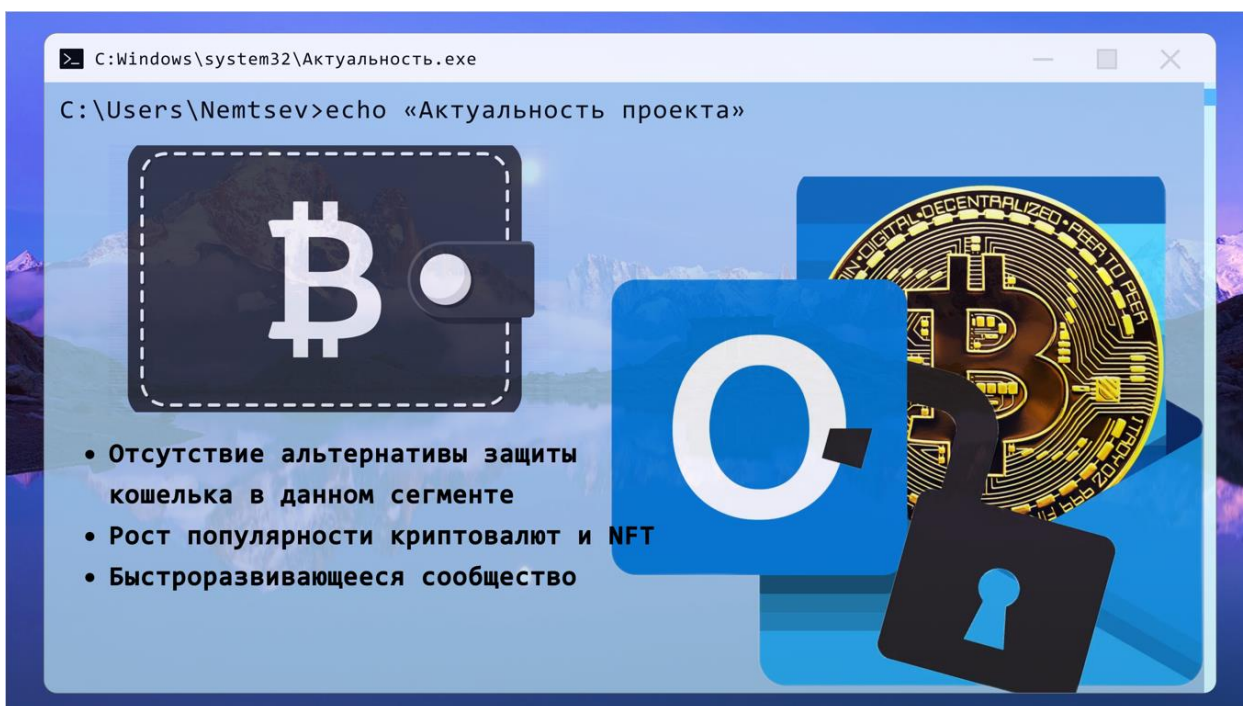
```

Приложение В

(Обязательное)

Система обезвреживания вредоносных программ несанкционированного доступа к криптокошелькам

Презентация

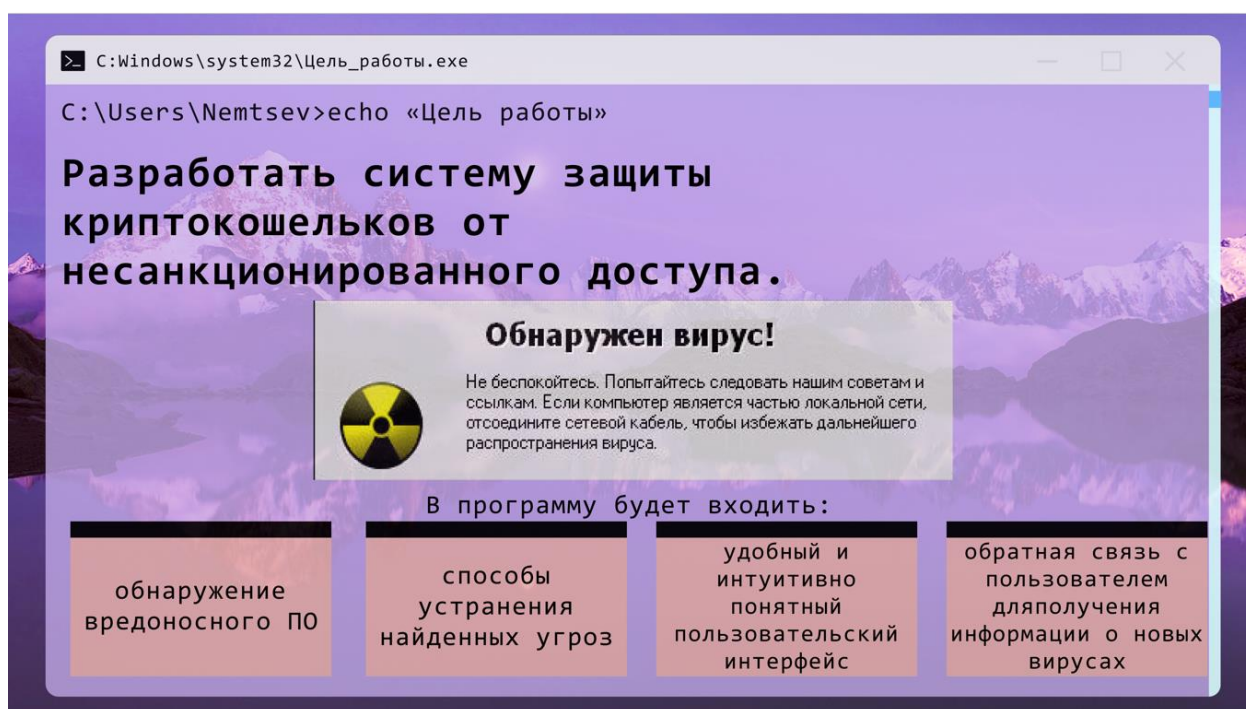
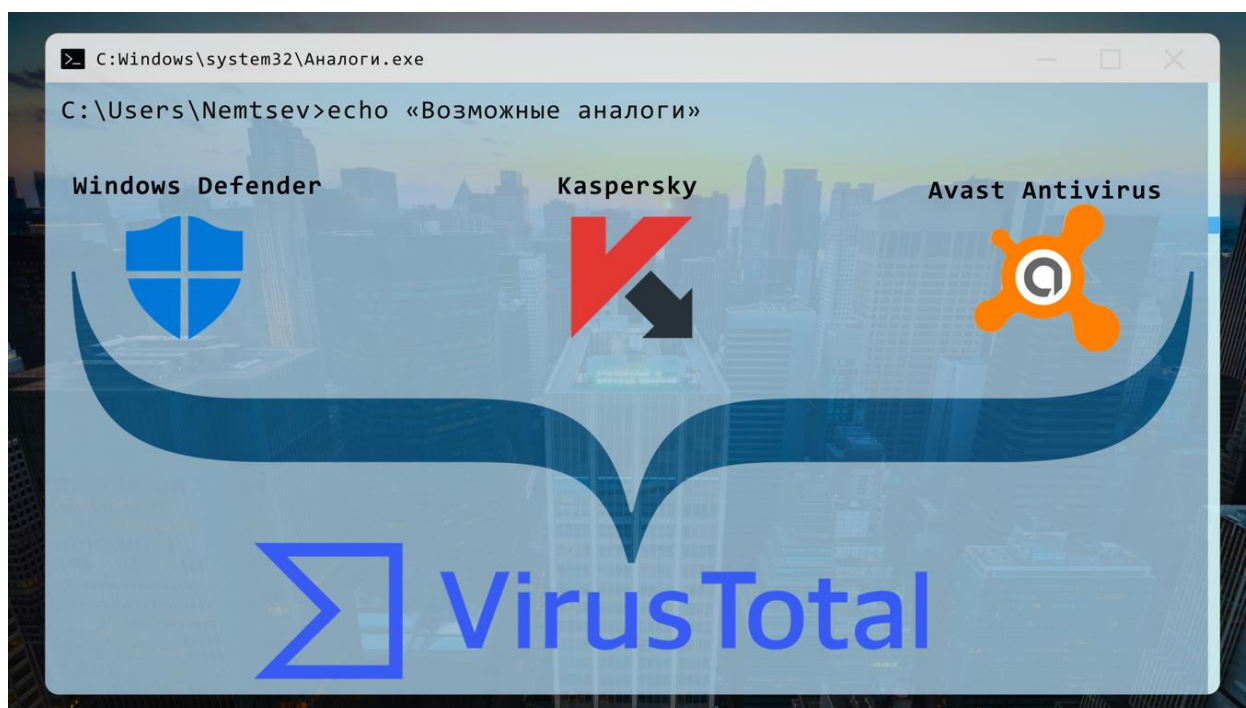


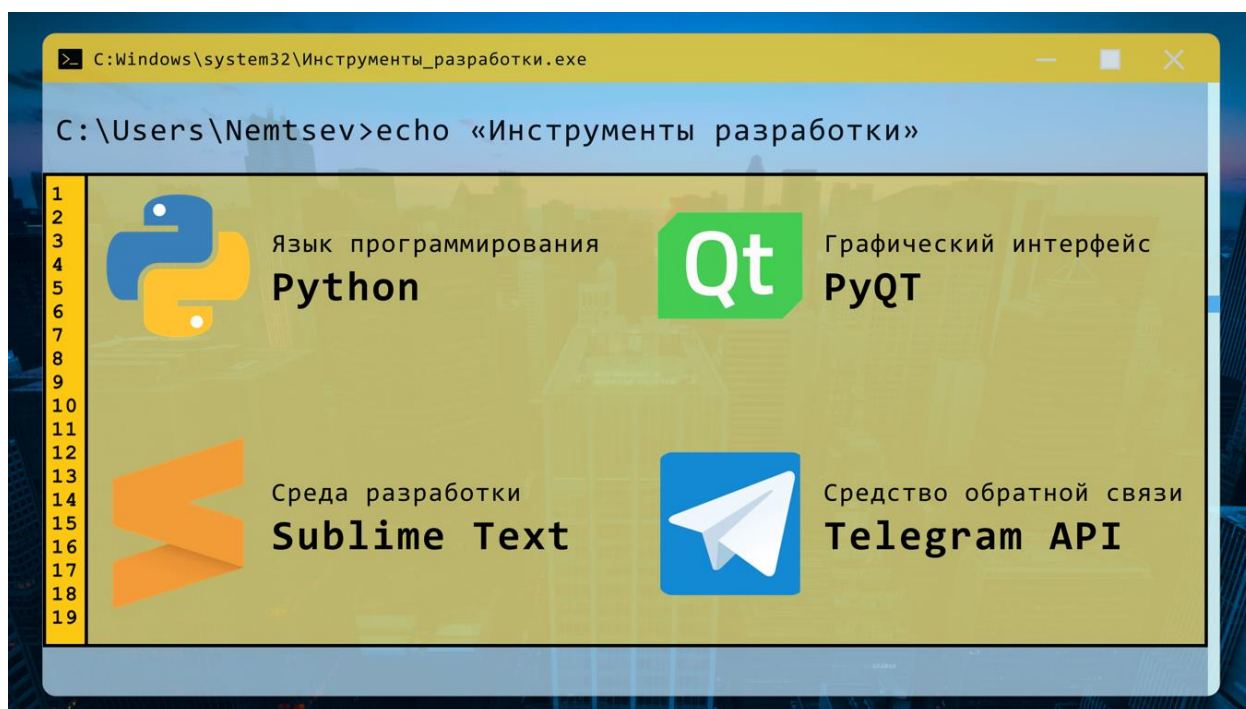
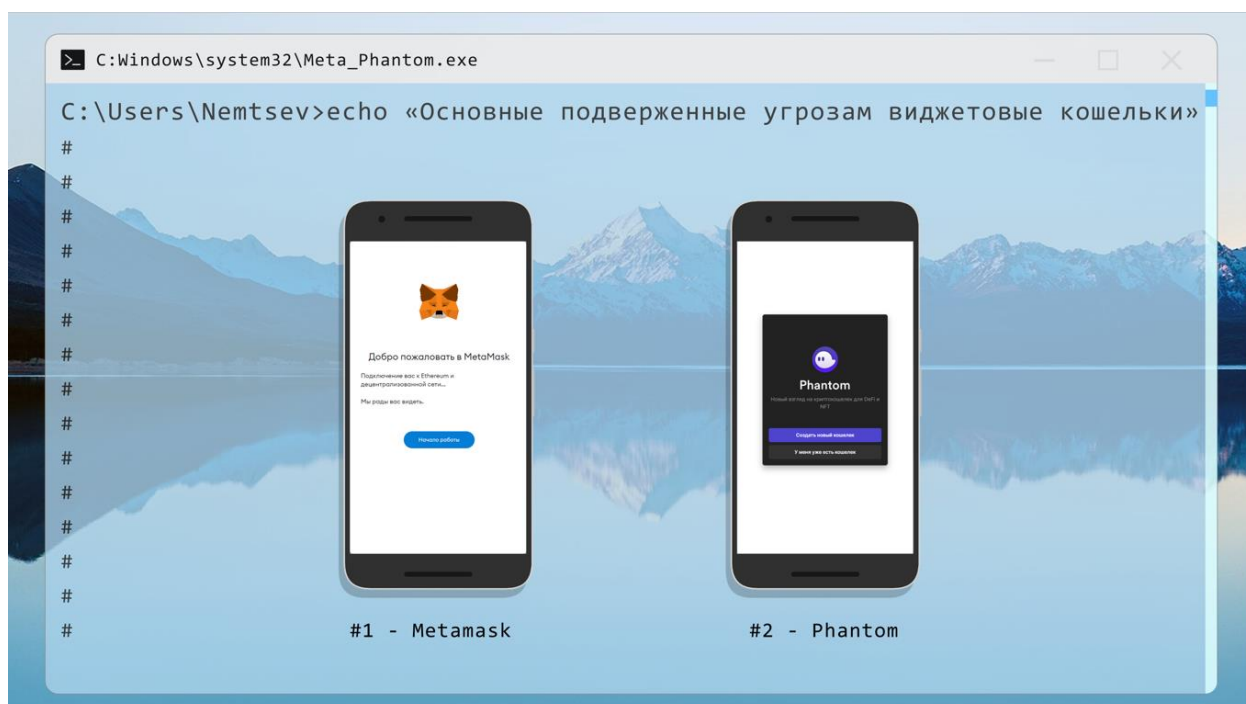
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.28.001 ПЗ

Лист

53



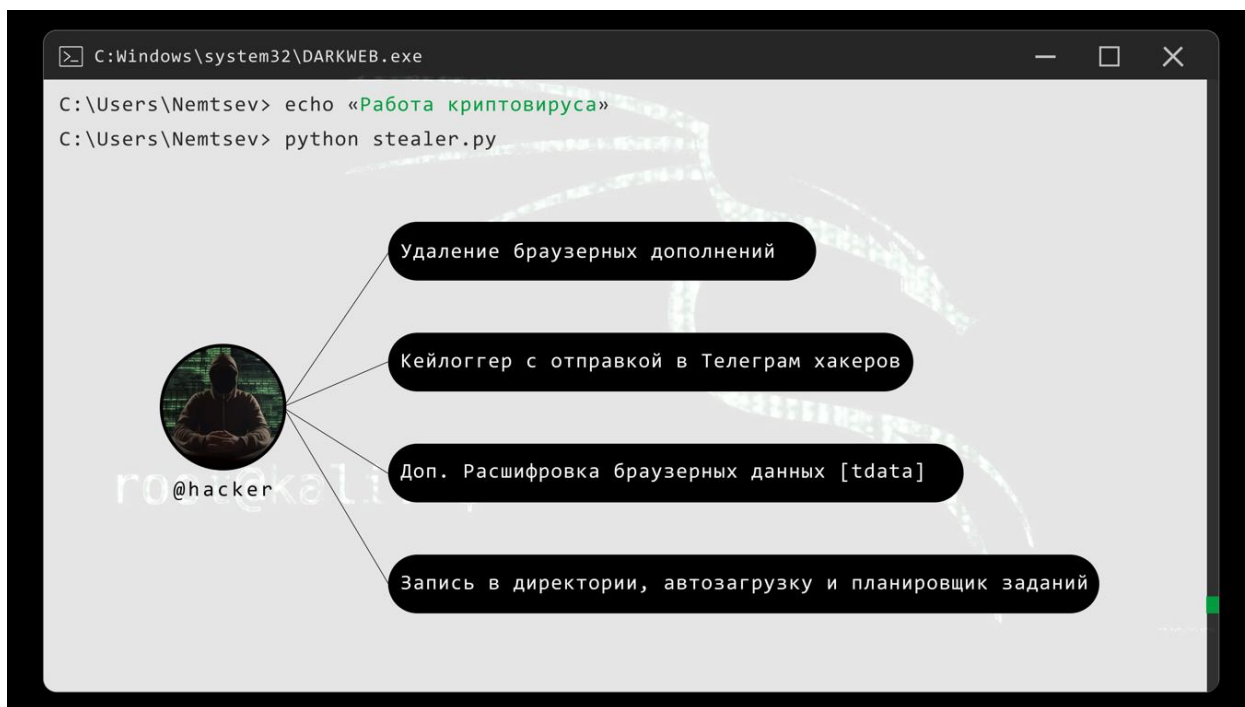
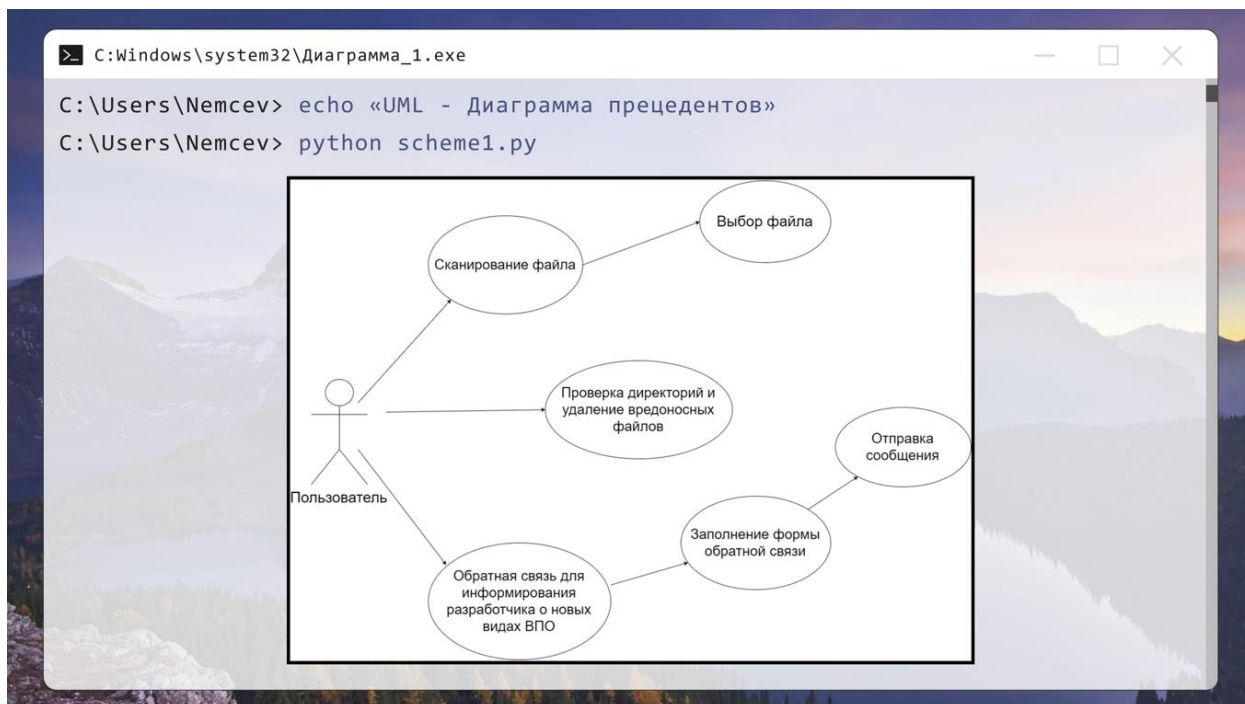


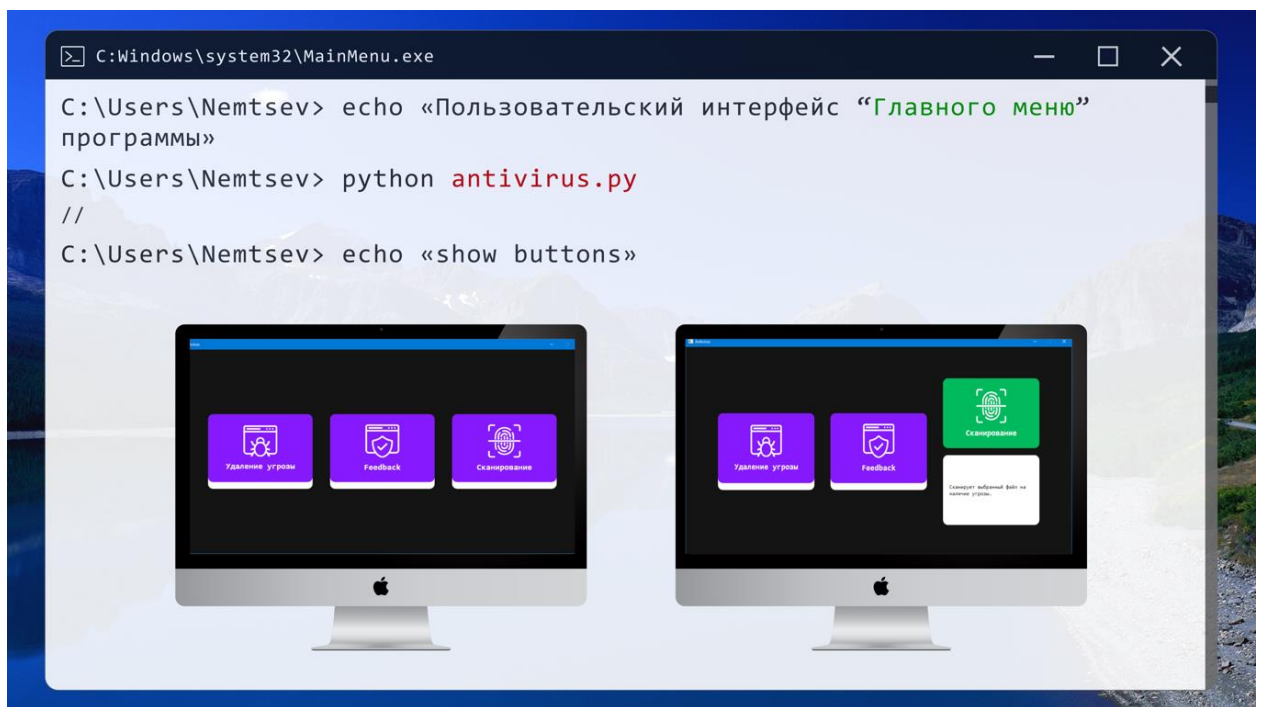
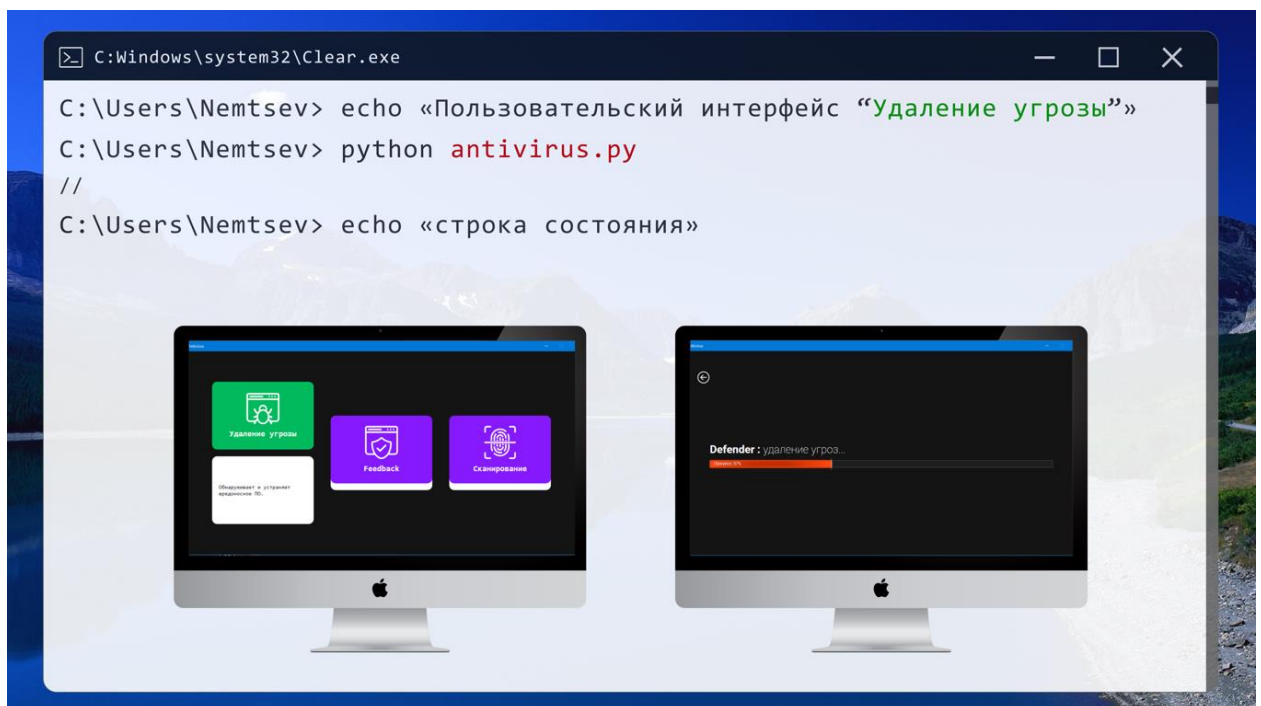
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.28.001 ПЗ

Лист

55





Изм.	Лист	№ докум.	Подпись	Дата

