МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-2 «Прикладные информационные технологии»

РАБОТА ДОПУЩЕНА К ЗАЩИТЕ

Заведующий
кафедрой _____О.В. Трубиенко

«___»_____20___ г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки (специальности) 10.05.04 _____
код направления подготовки

Информационно-аналитические системы безопасности
Наименование специальности

на тему: «Разработка сервиса поведенческого анализа функционирования мобильных устройств на операционных системах андроид»

Обучающийся _____ Князев Константин Антонович
подпись *фамилия, имя, отчество*

шифр 15Б0077

группа БИСО-02-15

Руководитель работы _____ к.т.н. доцент О.В. Трубиенко
подпись *ученая степень, ученое звание, должность* *фамилия, имя, отчество*

Консультант _____ к.т.н. доцент Г.Ю.Потерпеев
подпись *ученая степень, ученое звание, должность* *фамилия, имя, отчество*

Консультант по экономике _____ к.э.н. доцент Будович Л.С.
подпись *ученая степень, ученое звание, должность* *фамилия, имя, отчество*

Москва 2021 г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-2 «Прикладные информационные технологии»

СОГЛАСОВАНО	**УТВЕРЖДАЮ**
Заведующий кафедрой	Директор института
_____ *подпись*	_____ *подпись*
О.В. Трубиенко *фамилия, имя, отчество*	_____ *фамилия, имя, отчество*
«____» _____20___ г	«____» _____20___ г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы специалиста

Обучающийся Князев Константин Антонович
фамилия, имя, отчество

Шифр 15Б0077

Направление подготовки 10.05.04 *индекс направления* Информационно - аналитические системы безопасности *специальность*

Группа БИСО-02-15

1 Тема выпускной квалификационной работы специалиста

«Разработка сервиса поведенческого анализа функционирования мобильных устройств на операционных системах андроид»

2 Цель и задачи выпускной квалификационной работы специалиста

Цель работы: разработать и реализовать программное решение для реализации мониторинга устройств на операционной системе андроид.

Задачи работы:

1) провести анализ существующих ОС МПУ с последующим выбором целевой ОС;
2) выделить набор необходимых параметров для мониторинга;
3) провести анализ существующих средств мониторинга поведения мобильных устройств на выбранной операционной системе;
4) реализовать средство для поведенческого мониторинга мобильных устройств.

3 Этапы выпускной квалификационной работы специалиста

№ этапа	Содержание этапа ВКР специалиста	Результат выполнения этапа ВКР	Срок выполнения
1	Анализ современных мобильных операционных систем	выполнил	1.09.2020
2	Провести оценку возможностей существующих сервисов посредством метода анализа иерархий	выполнил	20.09.2020
3	Спроектировать систему мониторинга операционной системы мобильного персонального устройства с учетом выделенного набора параметров мониторинга	выполнил	12.10.2020
4	Реализовать собственную систему мониторинга операционной системы мобильного персонального устройства	выполнил	20.10.2020
5	Обоснование экономической эффективности решения	выполнил	25.11.2020

4 Перечень разрабатываемых документов и графических материалов

1) Пояснительная записка ВКР
2) Презентация

5 Руководитель выпускной квалификационной работы

Функциональные обязанности	Должность в Университете	Фамилия, Имя, Отчество	Подпись
Руководитель ВКР	Доцент, к.т.н.	Трубиенко О.В.	
Консультант по экономике	Доцент, к.э.н.	Будович Л.С.	

Задание выдал
Руководитель ВКР:
_____
подпись
«_01_» __сентября_____ 2020 г.

Задание принял к исполнению
Обучающийся:

_____
подпись
«_01_» __сентября_____ 2020 г.

АННОТАЦИЯ

Дипломная работа на тему: «Разработка сервиса поведенческого анализа функционирования мобильных устройств на операционных системах андроид».

Работа включает: 99 страниц, 16 рисунков, 20 таблиц, 13 приложений; использованных источников – 19.

Ключевые слова: поведенческий анализ, безопасность мобильных устройств, андроид, мониторинг защищенности.

Объект исследования – ОС компонентов мобильного сегмента ИТКС.

Предмет исследования – способы и методы контроля и поведенческого анализа мобильных персональных устройств.

Цель работы – разработать и реализовать программное решение для реализации мониторинга устройств на операционной системе андроид.

По результатам исследования предложен и разработан собственный сервис поведенческого анализа функционирования мобильных устройств на операционной системе андроид.

Полученный результат позволит контролировать и предупреждать действия пользователей корпоративных мобильных устройств, направленные на нарушение целостности и конфиденциальности корпоративной информации.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В данной работе используются следующие термины:

Вендор – это физическое или юридическое лицо, которое поставляет (а также, возможно, и производящее) объединенные в одну торговую марку товары и услуги.

Декомпиляция – процесс воссоздания исходного кода декомпилятором.

Дизассемблировоние – процесс восстановления ассемблерного кода.

Машинное обучение – класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач.

Нейросеть – представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и тому подобное.

Обфускация – запутывание кода. Приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

Прокси – это сервер и анонимайзер особого вида, представляющий собой веб-приложение, установленное на веб-сервере, выступающее в роли посредника.

Развертование – это все действия, которые делают программную систему готовой к использованию.

Слабосвязность – способ и степень взаимозависимости между программными модулями, при котором система хорошо структурирована и облегчает понимание логики модулей, их модификацию, автономное тестирование, а также переиспользование по отдельности.

Форензика – компьютерная криминалистика. Прикладная наука о раскрытии преступлений, связанных с компьютерной информацией, об

исследовании цифровых доказательств, методах поиска, получения и закрепления таких доказательств.

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

API – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

APK – формат архивных исполняемых файлов-приложений для Android, и ряда других операционных систем, основанных на Android.

BSSID – MAC-адрес точки доступа Wi-Fi.

Google Play – магазин приложений, а также игр, книг, музыки и фильмов от компании Google, позволяющий сторонним компаниям предлагать владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения.

HTTP – протокол прикладного уровня передачи данных.

IDE – комплекс программных средств, используемый программистами для разработки программного обеспечения.

JSON – текстовый формат обмена данными, основанный на JavaScript.

MAC – физическое имя устройства.

SSID – символьное название беспроводной точки доступа Wi-Fi.

SSL – криптографический протокол, который подразумевает более безопасную связь.

VPN – виртуальная частная сеть.

Wi-Fi – технология беспроводной локальной сети с устройствами.

Клиент-серверная архитектура – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

ПЕРЕЧЕНЬ СОКРАЩЕНИЕ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

В данной работе используются сокращения и условные обозначения, значения которых указаны в таблице 1.

Таблица 1 – Сокращения и условные обозначения

Сокращение	Значение
БД	База данных
ИТКС	Информационно-телекоммуникационная система
МПУ	Мобильное персональное устройство
ОЗУ	Оперативное запоминающее устройство
ОС	Операционная система
ПЗУ	Постоянное запоминающее устройство
ПК	Персональный компьютер
ПО	Программное обеспечение
API	Application programming interface
APK	Android package
BSSID	Basic Service Set Identifier
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
IDE	Integrated development environment
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
SMS	Short Message Service
SSID	Service Set Identifier
SSL	Secure Sockets Layer
VPN	Virtual Private Network

ВВЕДЕНИЕ

На текущий момент мобильные устройства тесно вплелись в жизнь каждого человека, став не просто голосовым средством связи с другим человеком, а, взяв на себя огромный пласт задач, которые раньше возлагались исключительно на персональные компьютеры. Возможным это стало из-за колоссального технического прорыва в сфере мобильных устройств, ведь сейчас они уже могут составить конкуренцию даже современным компьютерам.

Мобильное персональное устройство в наше время хранит огромные объемы данных, таких как переговоры, личные фотографии и файлы, текстовые сообщения, а также и платежные данные пользователей. А с постоянным ростом возможностей мобильных устройств увеличивается и число пользователей, и, как следствие, объемы, хранящейся на них, информации.

Большинство компаний используют современные технические средства для решения поставленных перед ними бизнес задач. Так, для решения большинства задач бизнеса мобильное устройство является наиболее удобным техническим средством для быстро взаимодействия с текстовыми документами, голосовой и письменной и видеосвязи с клиентами и коллегами. В связи с чем компании предоставляют своим сотрудникам доступ в корпоративную сеть для их персональных устройств, которые, зачастую, не оборудованы никакими средствами предупреждения и противодействия угрозам безопасности.

Однако в современной организации необходимо контролировать не только вредоносную активность компьютерных вирусов и вредоносного ПО, но и вредоносные или негативно влияющие на корпорацию действия, исходящие от самих пользователей устройств.

На рынке технических сервисов существуют системы и целые комплексы обеспечения безопасности, но большинство из них используют

средства и способы, сильно ограничивающие процесс использования мобильного устройства пользователем.

Все это делает мобильные устройства привлекательной целью для злоумышленников, желающих выкрасть денежные средства или получить доступ к персональным данным или коммерческим тайнам.

Ввиду этого, важным аспектом эксплуатирования подобных устройств является обеспечение их безопасности, мониторинг их функционирования с целью предотвращения разного рода утечек информации.

В рамках данной работы будут рассмотрены наиболее популярные на текущий момент операционные системы мобильных устройств. Будет проведен анализ существующих средств мониторинга и обеспечения безопасности мобильных устройств на операционной системе андроид. Приведена их критика, а также разработано и реализовано собственное решение данной проблемы современных мобильных устройств.

Актуальность данной работы заключается в том, что большинство мобильных устройств на сегодняшний день не обладают дополнительными средствами защиты их целостности и конфиденциальности хранящихся на них данных. А корпоративной сфере, где мобильные устройства являются основным техническим средством решения поставленных задач, требуют более детального контроля безопасности.

Цель работы – разработать и реализовать программное решение для реализации мониторинга устройств на операционной системе андроид.

Задачи работы:

1) провести анализ существующих ОС МПУ с последующим выбором целевой ОС;

2) выделить набор необходимых параметров для мониторинга;

3) провести анализ существующих средств мониторинга поведения мобильных устройств на выбранной операционной системе;

4) реализовать средство для поведенческого мониторинга мобильных устройств.

Объект исследования – ОС компонентов мобильного сегмента ИТКС.

Предмет исследования – способы и методы контроля и поведенческого анализа мобильных персональных устройств.

В первой главе будет рассмотрена актуальность проблемы защищенности мобильных устройств. Проанализированы наиболее популярные операционные системы для МПУ и обоснован выбор целевой операционной системы.

Во второй главе будут рассмотрены существующие методы анализа безопасности мобильных персональных устройств, исследованы существующие решения проведения поведенческого анализа мобильных устройств на ОС андроид, выделены их слабые и сильные стороны, на основе которых будут составлены необходимые критерии обеспечения безопасности, предъявляемые к системам мониторинга и предупреждения ее нарушения.

Во третьей главе работы будет выбран необходимый набор технологий, разработано и реализовано средство проведения поведенческого анализа функционирования мобильных устройств на операционных системах андроид, альтернативное существующим.

В четвертой главе будут рассчитаны показатели экономической эффективности при внедрении разработанного программного обеспечения.

Новизна работы заключается в разработке собственного решения, способного составить конкуренцию имеющимся средствам защиты и мониторинга целостности мобильных устройств, устранив частично или полностью недостатки существующих средств.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Актуальность проблемы

На сегодняшний день мобильные устройства стали неотъемлемой частью каждого человека, с помощью которых решаются задачи средств связи, передачи данных, навигации и прочего.

Обратимся к данным известной по всему миру компании в сфере it-аналитики, проводящей исследования веб-трафика – StatCounter [1] и проанализируем статистику, представленную на рисунке 1.1, то начиная с 2012 года замечен резкий рост популярности мобильных устройств и такой же резкий спад популярности персональных компьютеров.

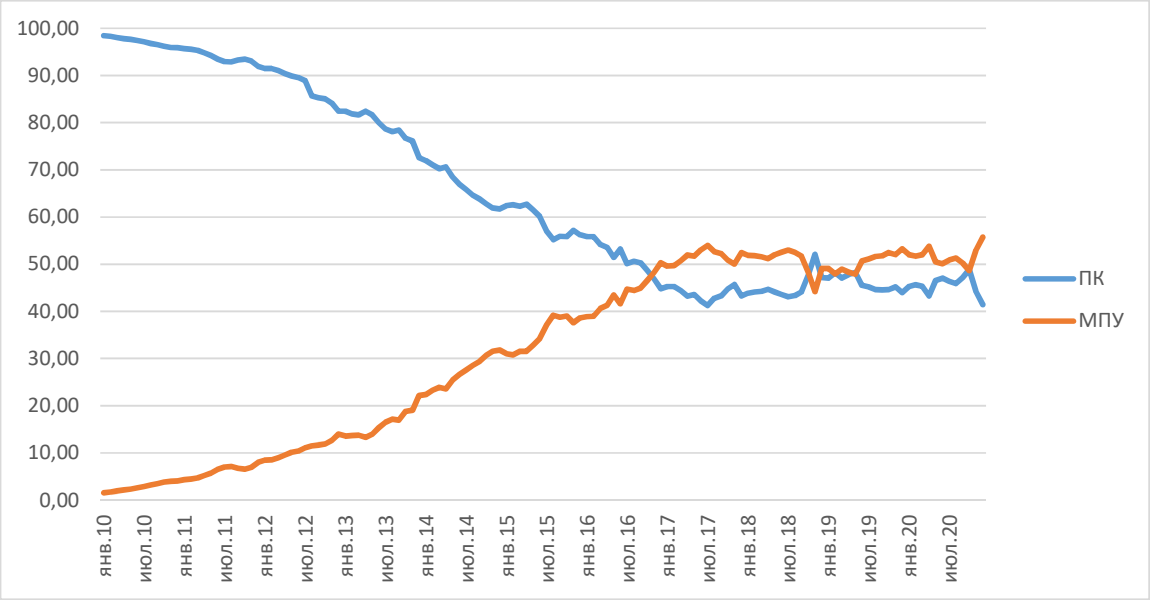

Рисунок 1.1 – Темпы роста популярности ПК и МПУ

Как можно заметить уже в 2017 году мобильные устройства стали более популярными, чем персональные компьютеры и, несмотря на периодический рост и спад обоих типов устройств, заметна тенденция к увеличению популярности именно мобильных устройств.

Вызвана такая динамика тем, что большинство мобильных устройств уже много лет справляются с задачами, которые были всегда возложены на персональные компьютеры, ничуть не хуже, как, например, редактирование

текстовых файлов, а иногда даже лучше, чем компьютеры.

В добавок неоспоримым преимуществом мобильных устройств является простота их транспортировки, ввиду их веса и габаритов.

Если рассмотреть актуальное состояние популярности этих устройств, то на рисунке 1.2 видно, что мобильные устройства не перестают уступать в популярности современным компьютерам и остаются на лидирующих позициях.

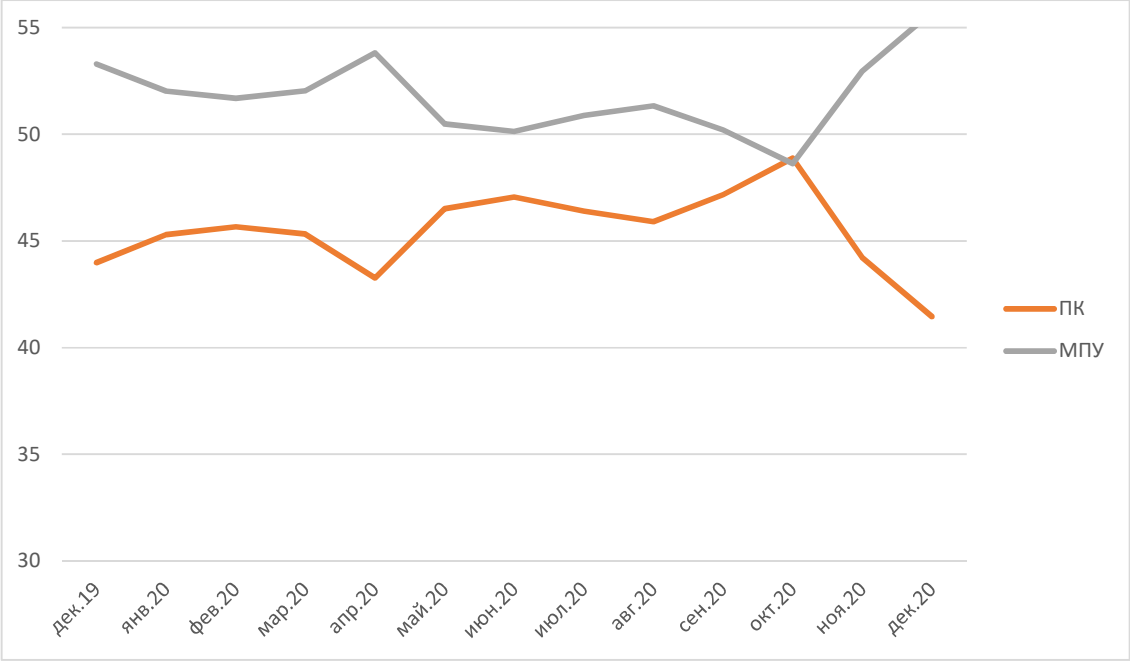

Рисунок 1.2 – Темпы роста популярности мобильных устройств и персональных компьютеров в 2020 году

И если с угрозами безопасности персональных компьютеров люди столкнулись еще в прошлом веке, то с мобильными устройствами ситуация обстоит гораздо хуже. Большинство средств предотвращения угроз ИБ направлены на компьютеры и сейчас существует лишь немного компаний, предоставляющих ПО для сохранения целостности информации и ее конфиденциальности на мобильных устройствах, при том, их нельзя назвать популярными.

Хоть и целевой аудиторией использования мер обеспечения ИБ МПУ являются корпорации, с намерением контроля целостности и

конфиденциальности коммерческой информации, стоит отметить что немаловажным является обеспечение контроля действий самого пользователя, то есть сотрудника, который может оказаться инсайдером и по своей воле или по незнанию передать данную информацию злоумышленнику.

Вместе с тем, существует вероятность утечки не только коммерческой информации, но и личной информации обычного пользователя, который тоже заинтересован в сохранности информационной безопасности своего устройства и своих данных.

С целью обеспечения ИБ существует два типа анализа вредоносного ПО: статистический и динамический (поведенческий) [2]. Суть динамического анализа заключается в отслеживании разнообразных сведений системы в момент работы приложения с целью обнаружения нелегетимных действий. Статический, напротив, подразумевает поиск возможных уязвимостей в исходном коде рассматриваемого приложения. Более подробно эти методы будут рассмотрены далее.

На текущий момент уже существуют средства поведенческого анализа мобильных приложений. Но преимущественно они предназначены для этапа тестирования приложений перед предоставлением их пользователю.

Наиболее эффективным средством обеспечения безопасности мобильных устройств является динамический анализ функционирования ОС, так как такие устройства практически всегда находятся в использовании, поэтому для контроля ИБ для мобильных устройств наиболее актуальным является проведение поведенческого анализа в режиме реального времени.

Рассмотрим задачу на примере динамического анализа работы гипотетического банковского приложения.

Если обратиться к статистике, составленной компанией Positive Technologies за 2016 год, в банковских приложениях существуют недостатки реализации двухфакторной аутентификации у 71%, а 33% содержат уязвимости, позволяющие украсть деньги. [3]

Для решения этой проблемы, многие компании используют разные

способы анализа защищенности своего приложения. Так, например, компания Smart Security предлагает использовать метод поведенческой биометрии, который основан на поведенческом анализе действий системы от взаимодействия с пользователем (различные жесты, скорость ввода пин-кода и прочих). Всю собранную от анализа статистику обрабатывает сервер с помощью технологий нейросетей и машинного обучения, благодаря чему сервер понимает, легитимный ли пользователь пользуется приложением или доступ к нему получило стороннее лицо [4].

1.2 Анализ современных мобильных ОС

Мобильные устройства начали развиваться заметно позже стационарных компьютеров и ноутбуков. Но со временем такие устройства, как мобильные телефоны и планшеты становились все мощнее и совершеннее, а это означает, что для них появилась необходимость в соответствующем программном обеспечении.

Явным различием в мобильных ОС и операционных систем для ПК – необходимость в таких средствах связи, как звонки, смс и выход в интернет.

Огромное количество вендоров занимались изобретением своей ОС для мобильных устройств, но не обладали достаточными конкурентными преимуществами по сравнению с системами, которые будут рассмотрены далее. Поэтому такие системы не приобрели широкого распространения и в данной работе рассматриваться не будут.

Проанализируем темп роста популярности ОС для ПК и МПУ, чья статистика распространения со временем представлена на рисунке 1.3.

Можем обратить внимание, что с середины 2017 года, операционная система андроида обогнала по популярности абсолютного лидера среди ОС – Windows. Второй мобильной операционной системой, имеющей заметную популярность, является iOS.

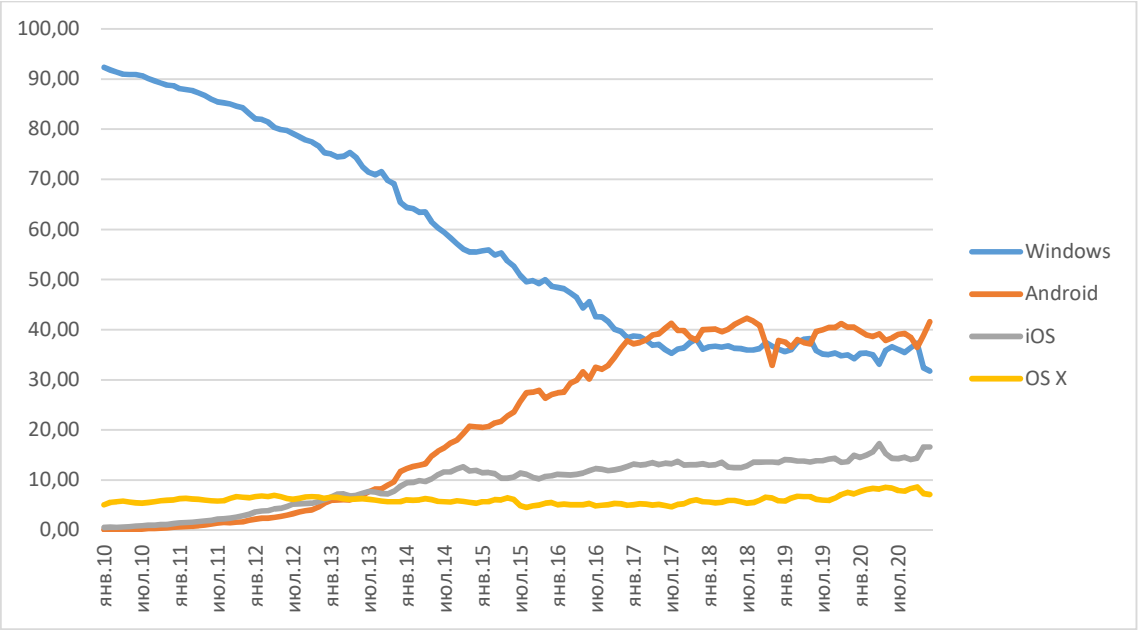

Рисунок 3 – Темпы роста популярности операционных систем

Если рассмотреть статистику за 2020 год, то на рисунке 1.4 отметим, что абсолютным лидером среди как мобильных, так и вообще операционных систем является Android, а iOS почти в 2,5 раз менее популярная.

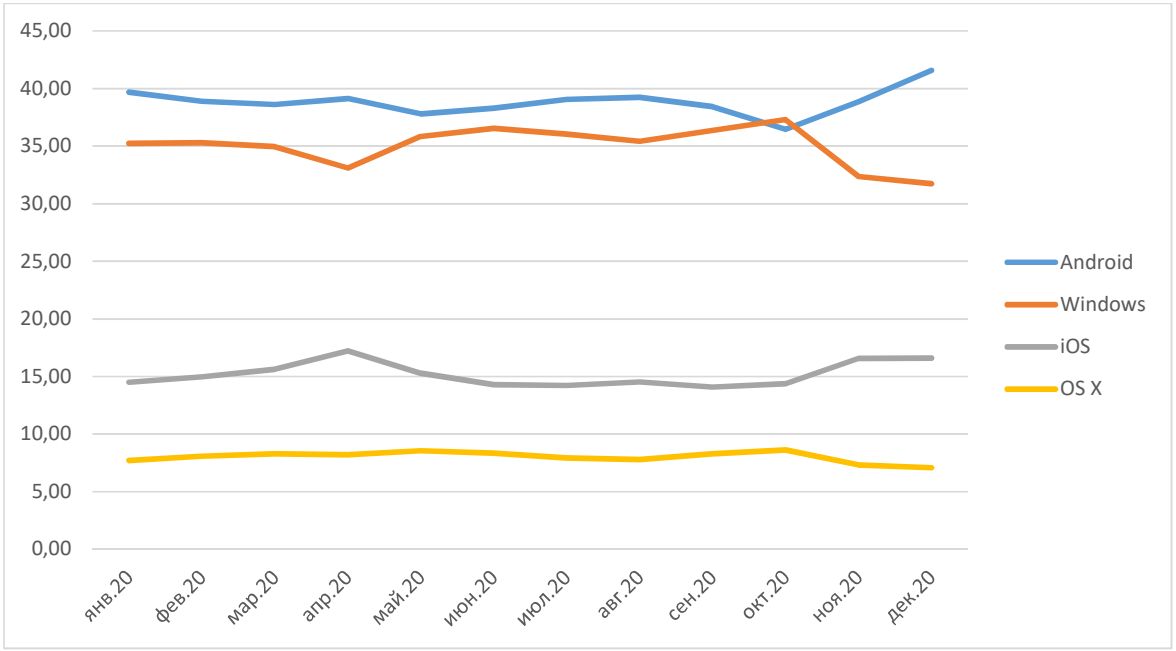

Рисунок 4 – Темпы роста популярности операционных систем в 2020 году

Таким образом можно выделить две наиболее популярные мобильные ОС – Android и iOS. Для расширения списка анализируемых систем, возьмем к рассмотрению мобильную ОС от компании Microsoft – Windows Phone.

Хотя и по сей день, многие вендоры продолжают оснащать свои устройства собственной ОС, но зачастую, их ОС построена частично или полностью на представленных выше системах.

1.3 Android

Android – операционная система, разрабатываемая компанией Android Inc., которая была выкуплена компанией Google в июле 2005 года [5]. Операционная система основана на ядре Linux и виртуальной машины Java, созданной Google. Впервые эта система была представлена миру в сентябре 2008 года и с тех пор сменила десяток своих версий.

Компания Google в ноябре 2007 года объявила ОС Android открытой для использования сторонними компаниями. Ввиду этого, как было сказано ранее, многие компании создают свои версии операционных систем для своих устройств, дополняя доступную им версию данной мобильной операционной системы собственными функциями, возможностями и собственным пользовательским интерфейсом. Именно поэтому данная ОС используется для большинства смартфонов, планшетов, электронных книг, цифровых проигрывателей, умных часов, умной бытовой технике и прочего.

Опираясь на статистику, представленную на рисунке 1, Android является наиболее популярной операционной системой в мире, составляя примерно 40% из всех используемых ОС, включая системы для ПК такие, как Windows [6]. Ввиду этого фактора можно сказать, что на текущий момент это самая популярная операционная система.

К основным минусам системы можно отнести то, что несмотря на непрерывную работу компании Google над своей ОС, в частности, регулярных обновлений, связанных с безопасностью системы и хранения данных, система, ввиду доступности своего исходного кода, считается наиболее взламываемой системой.

1.4 iOS

iOS – мобильная операционная система для смартфонов, планшетов, портативных проигрывателей и некоторых других устройств, которая была разработана, выпускается и поддерживается компанией Apple с июня 2007 года. А в 2014 году появилась поддержка автомобильных систем Apple CarPlay [7].

В ней используется ядро XNU, которое тоже производится компанией Apple. В отличии от Android, данная ОС выпускается только для устройств, создаваемых компанией Apple, что значительно снижает возможную нагрузку на ОС, связанную с разнообразием оборудования (процессоры, оперативная память и тд).

В определенный момент, компания Apple начала делить свою мобильную ОС, оставляя работоспособность iOS, в основном для своих смартфонов, а для других гаджетов начала создавать новые операционные системы.

Так, например, в апреле 2015 года появилась WatchOS – операционная система для устройств Apple Watch, основанная на iOS, а в сентябре 2019 года была выпущена iPadOS, предназначенная планшетов Apple iPad.

По статистике на 2020 год, все семейство операционных систем, основанных на iOS занимает примерно 15% среди всех используемых ОС в мире.

В противовес мобильной операционной системы от Google, исходный код каждой из семейства iOS не является общедоступным, по этой причине принято считать, что если на устройстве не стоит Jailbreak (root-доступ), то система считается одной из наиболее защищенных от злоумышленников.

1.5 Windows Phone

Windows Phone – одна из версий операционных систем, созданных компанией Windows.

До этого эта компания создавала операционную система Windows Mobile, которая была представлена миру в далеком 2000 году и перестала поддерживаться своим производителем в феврале 2010 года, объявив Windows Phone приемником своей мобильной ОС.

В сравнении с выше рассмотренными операционными системами, данная ОС не смогла составить конкуренцию, из-за чего стала менее распространена в сравнении с конкурентами.

В марте 2015 года была представлена последняя версия данной ОС, в то время как следующее поколение мобильных ОС от Windows было уже представлено миру под именем Windows 10 Mobile. Но уже в 2018 году была прекращена активная работа над этим проектом, а в январе 2020 года была окончательно прекращена ее поддержка [9].

Возвращаясь к статистике используемых ОС в мире, по состоянию на 2020 год, мобильные ОС от Windows входят в набор операционных систем, чье распространение составило меньше 3%.

В связи с вышесказанным, семейство ОС Windows не интересно для рассмотрения в рамках данной работы.

Выводы первого раздела

Ввиду доступности системы, многие компании производят мобильные устройства на ОС Android, что в свою очередь рождает большую конкуренцию на рынке мобильных устройств. В связи с чем, чаще всего в качестве корпоративных устройств связи в компаниях используют мобильные ОС Android с целью сокращения расходов, в сравнении с устройствами от компании Apple.

По итогу анализа наиболее популярных мобильных ОС можно сделать вывод, что целесообразно, ввиду своей распространенности, выбрать в качестве целевой ОС Android для поиска дополнительных средств обеспечения информационной безопасности.

2 СПЕЦИАЛЬНЫЙ РАЗДЕЛ

2.1 Принципы анализа следов вредоносного ПО для ОС Android

Сама по себе система андроид состоит из набора приложений, каждое из которых выполняет ту или иную функциональность системы в целом.

Большинство приложений для работы в ОС Android написано на языке программирования Java. Внутри системы приложения выполняются с помощью AndroidRuntime, начиная с Android 4.4.

Приложения хранятся в памяти мобильного устройства, упакованные в файл формата APK, предварительно скомпилированные. Данный файл представляет собой архив, содержащий байт-коды, ресурсы, сертификаты и файл manifest. После установки приложения APK-файл копируется в файловую систему, обычно для системных приложений это каталог /system/app, а для пользовательских – /data/app. [9]

С точки зрения информационной безопасности, наиболее важными части APK-файла является его сигнатура, байт-код и ресурсы.

Различают два типа анализа вредоносного ПО:

– Статический;

– Динамический (поведенческий).

2.1.1 Статический анализ вредоносного ПО для андроид

В рамках статического анализа вредоносного ПО производится анализ его кода, где главной задачей является выделение фрагментов кода, содержащие предположительно вредоносные действия.

Для статического анализа приложений для Android обычной используют такие средства, как ApkTool, Dex2Jar.

Программа ApkTool позволяет дизассемблировать программу, в результате которой будут получены:

– Файл AndroidManifest.xml (в котором содержатся сведения о запрашиваемых приложением разрешениях, а также указаны точки входа);

– Каталог ресурсов (в котором содержатся описанные в формате XML макет приложения и графичекие его элементы);

– Каталог операционного кода (содержащий файлы с расширением *.smali*, которые также могут быть использованы для анализа на присутствие потенциально вредоносного кода).

Программное средство Dex2Jar позволяет преобразовать байт-код в код на языке Java. Для этого необходимо распаковать APK-файл приложения и обработать в программе Dex2Jar файл class.dex, извлеченный из APK. После чего, с помощью таких средств как JS-GUI или аналогичных, произвести анализ Java кода.

2.1.2 Динамический анализ вредоносного ПО для андроид

Обычно, для проведения динамического анализа прибегают к использованию вспомогательных средств, собирающих информацию, а иногда и статистику, функционирования приложения и системы во время работы приложения [9].

Для примера, динамический анализ потенциально вредоносного приложения для Android с помощью программы DroidBox.

Данная программа позволяет получить:

– хеш-сумму APK-файла (прибегая к алгоритмам хеширования MD5, SHA-1, SHA-256),

– сведения о полученных и отправленных по сети данных,

– сведения об операциях чтения и записи файлов,

– сведения о запущенных службах и загруженных классах,

– сведения о сборе и отправке пользовательских данных,

– сведения о разрешениях, которое получило приложение,

– сведения о криптографических операциях, производимых приложением с помощью Android API,

– сведения об отправляемых SMS-сообщениях и осуществляемых вызовах.

В результате работы программы будет получен файл в формате JSON, содержащий вышеперечисленные сведения.

2.2 Этапы анализа следов предположительно вредоносного ПО в ОС Android

Подводя итог по описанным методам анализа приложения для Android, можно составить схему поэтапного проведения анализа программы с целью обнаружения следов вредоносного ПО:

1) исследование установленных в системе приложений,
2) идентификация потенциально вредоносной программы,
3) преодоление мер противодействия криминалистическому анализу ПО,
4) динамический анализ программы,
5) Статический анализ программы,
6) выделение потенциально опасных фрагментов кода.

2.3 Обнаружение вредоносных приложений для Android

Существуют специализированные средства для обнаружения вредоносных программ, некоторые из которых будут рассмотрены далее, однако они не решают проблему форензики, так как основной целью подобного анализа является не только выявления потенциального вредоносного ПО, но и проверка его кода и реконструкция действий злоумышленника с целью поиска способов устранения уязвимости.

Для обнаружения потенциального вредоносного приложения, эксперт может использовать контрольные суммы. Вычисленную контрольную сумму можно сверить с контрольной суммой, предоставляемой поставщиком приложения или сверив ее с базой данных контрольных сумм сервиса Google Play. В случае несовпадения контрольных сумм, можно предварительно предположить, что приложение является потенциальным вредоносным ПО.

Однако данный метод не дает окончательного результата экспертизы. Для заключения результата необходим более глубокий анализ.

Для криминалистического исследования ресурсов, наибольший интерес представляет файл AndroidManifest.xml, так как он содержит сведения о запрашиваемых приложением разрешений, будь то доступ к микрофону, геолокации или списку контактов. Соответственно, изучение списка запрашиваемых разрешений является одним из первых шагов криминалистических тактик.

Стоит отметить, что приложению достаточно единожды получить разрешение. Нередко пользователи бездумно дают приложению запрошенные разрешения при первом его запуске. Так, на первый взгляд, безобидное приложение может получить доступ к файловой системе, списку контактов и так далее. Таким образом, подозрительные для приложения разрешения – один из ключевых признаков вредоносного ПО.

2.4 Методы, затрудняющие криминалистический анализ

Существует четыре наиболее распространенных метода противодействия криминалистическому исследованию:

- обфускация,
- шифрование символьных строк,
- противодействие декомпиляции,
- проверка окружения.

2.4.1 Обфускация

Для деобфускации кода необходимо предварительно произвести декомпиляцию приложения. На текущий момент не существует идеальных средств декомпиляции. Нередко получаемый с их помощью код не полон и содержит ошибки. Однако в случае как с приложениями для Android, так как байт-код всегда точен, необходимо опираться не только на

декомпилированный код, но и сам байт-код программы.

Для извлечения байт-кода из APK-файла можно воспользоваться готовыми средствами, например, ApkTool. После данной операции необходимо также воспользоваться существующими уже средствами, как Dex2Jar, который преобразует извлеченный байт-код в код на языке Java. После этого можно приступить к деобфускации исходного кода приложения.

2.4.2 Шифрование символьных строк

Символьные строки, содержащиеся в программе, являются важным источником криминалистически значимой информации. Для шифрования символьных строк обычно используют алгоритмы XOR, Base64 или более сложные алгоритмы, например, DES, AES и тому подобные.

При расшифровке стоит учитывать, что для ее усложнения строки могут быть последовательно зашифрованы комбинациями различных алгоритмов.

2.4.3 Проверка окружения

Встречаются вредоносные ПО, которые функционируют исключительно на определенных устройствах. В таком случае приложение проверяет свойства системы и выполняет свой вредоносный код, если свойства соответствуют заложенным в программу.

Таким образом, программа не будет работать на устройстве или эмуляторе, не проходящим эту проверку. Данный метод применяется для затруднения динамического анализа.

Обойти данный метод можно, если модифицировать код обнаруженной программы.

2.5 Выделение каналов утечки информации

В рамках использования МПУ в корпоративных целях, помимо проведения анализа на наличие вредоносного ПО функционирующего в системе андроид, следует также обратить внимание на контроль легитимности

действий пользователя, с целью предотвращения утечек информации.

Если обратиться к статистике от компании InfoWatch за три первых квартала 2020 года (рисунок 2.1), наиболее популярным каналом утечки корпоративной информации является инсайдер, обычно, сотрудник не непривилегированный доступом к утекаемой информации.

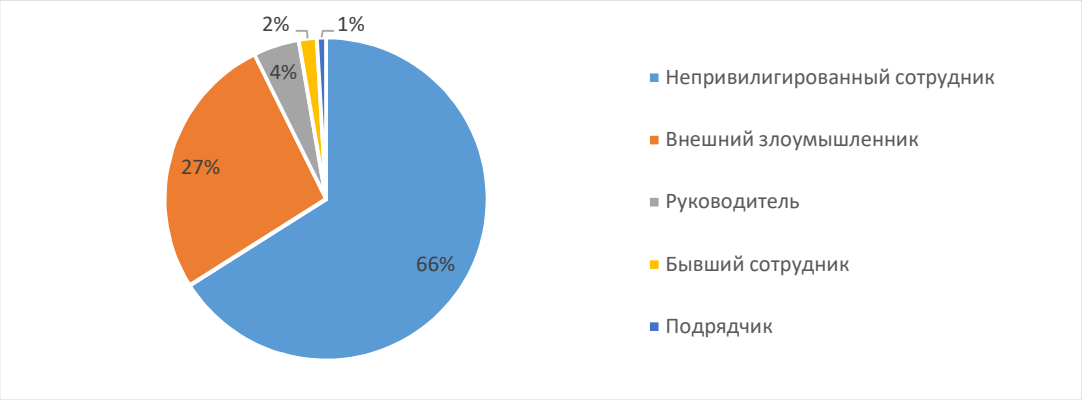

Рисунок 2.1 – Статистика источников утечки информации в компаниях

Для устранения подобных инцидентов, необходим мониторинг сотовой мобильной связи и контроль обмена информацией по средствам смс, что может привести к обнаружению инсайдера и попытке предотвратить несанкционированный доступ.

Помимо нарушения ИБ самим инсайдером, существует не менее популярный способ ее нарушения – социальная инженерия, с помощью которой внешний нарушитель может так же получить доступ к секретной информации.

С целью предотвращения нарушения ИБ, необходимо анализировать интернет трафик, в частности, трафик почтовых служб и мессенджеров.

Помимо прочего, мониторинг интернет трафика поможет проанализировать передачу данных и выявить факт несанкционированного доступа к мобильному устройству, который может быть не только физическими каналами, рассмотренными ранее, но и с помощью технических каналов, которые становятся доступны злоумышленнику после его контакта с атакуемой системой, после чего целостность и конфиденциальность данных

может быть нарушена по акустическим, вибрационным, электрическим и прочим каналам.

2.6 Обзор средств проведения динамического анализа приложений для Android

Наибольший интерес, ввиду объема закрытых к распространению данных, представляют мобильные устройства, используемые в коммерческих целях.

Рассмотрим примеры сервисов, которые позволят в момент работы приложения в ОС Android определить легитимность производимых действий или обнаружат утечку данных, а также рассмотрим плюсы и минусы этих сервисов.

Средства динамического анализа отслеживают поведение неизвестных приложений во время выполнения, выполняя целевое приложение для создания поведенческого следа. Динамический анализ может отслеживать поведение приложения с использованием одного (или нескольких) из следующих методов:

– Отслеживание заражения: инструменты отслеживания заражения часто используются в структурах динамического анализа для реализации общесистемного динамического распространения заражения с целью обнаружения потенциального злоупотребления личной информацией пользователей;

– Самоанализ виртуальных машин (VMI): инфраструктуры на основе VMI [10] перехватывают события, происходящие в эмулируемой среде. Системы на основе Dalvik VMI отслеживают выполнение Android API посредством модификаций в Dalvik VM;

– Мониторинг системных вызовов: платформы могут собирать обзор выполненных системных вызовов, используя, например, VMI или модуль ядра. Это позволяет (частично) отслеживать собственный код;

– Трассировка методов: платформы могут отслеживать вызовы Java-методов приложения в Dalvik VM;

Популярный фреймворк для отслеживания заражений – TaintDroid [11]. TaintDroid реализован на виртуальной машине Dalvik и мониторах заявки на утечку конфиденциальной информации. Однако, ScrubDroid [12] представил ряд атак для обхода динамического анализа заражения. VetDroid [13] – это динамический фреймворк, который измеряет фактическое поведение при использовании разрешений динамически построение графика использования разрешений. Их метод заражения данных построен на TaintDroid, но улучшает его, определяя неявные точки использования явных разрешений.

Очевидно, что на смартфоне не получится полноценно анализировать подозрительные приложения в связи с ограниченными ресурсами самого устройства. Простое сканирование файлов займёт немало времени и «съест» заметную долю заряда аккумулятора. Одним из способов решения этой проблемы является анализ с помощью облачных сервисов.

На рисунке 2.2 изображена схема работы Crowdroid [14], инструмента для анализа поведения приложений. Это приложение, которое производит мониторинг системных вызовов, инициированных наблюдаемым приложением, предварительно обрабатывает их и отправляет в облако. После чего с помощью кластеризации определяет, является ли приложение вредоносным или нет. Алгоритм кластеризации, применяет метод k-средних, чтобы различать безобидные приложения и соответствующую им версию вредоносного ПО. Что касается механизма связи между клиентом Crowdroid и сервером, он осуществляется с использованием протокола FTP, без акцента на защите конфиденциальности передаваемых данных. Если злоумышленник обнаруживает и манипулирует трафиком в процессе связи, это может привести к ошибкам неправильной классификации.

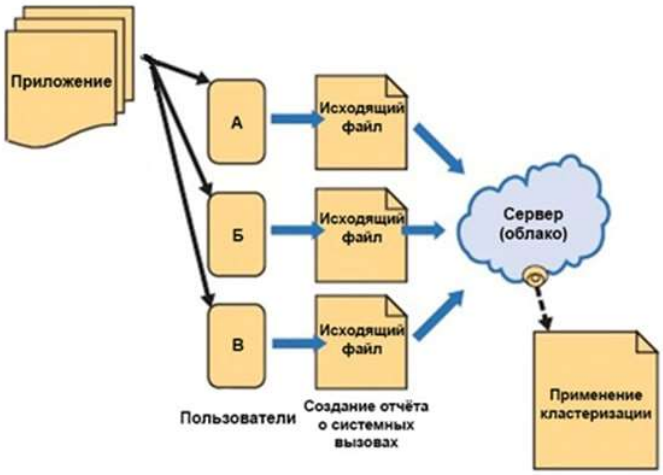

Рисунок 2.2 – Схема работы Crowdroid

К сожалению, у Crowdroid бывают ложно положительные результаты, если данных для анализа очень мало. Также к недостаткам можно отнести следующее: во-первых, система всегда разделяет векторы данных системных вызовов на два кластера, даже если в ней нет вредоносных программ. Кластерное отображение будет кардинально меняться всякий раз, когда вредоносный вектор выполнения входит в набор данных, что усложняет процесс анализа и затрудняет работу МПУ. Перечисленные проблемы требуют некоторой ручной проверки или дальнейшего автоматического анализа. Во-вторых, можно намеренно отправить правильные данные в систему, оставив набор данных поврежденным.

Для определения наиболее эффективного решения для проведения поведенческого анализа мобильных устройств на операционной системе андроид, проведем иерархический анализ некоторых из них.

Выделим факторы, которые будут являться критериями сравнения исследуемых сервисов:

– возможность проведения непрерывного анализа в режиме реального времени (К1),

– автоматизированный запуск (К2),

– анализ действий пользователя (К3),

– анализ на наличие вредоносного ПО (К4),

– нейтральность для антивирусных систем (К5),

– простота использования (К6),

– устойчивость к нестабильному соединению (К7),

– доступность (К8),

– корректность работы (К9).

В качестве исследуемых альтернатив анализа функционирования мобильных устройств на ОС андроид выберем следующие сервисы:

– Crowdroid,

– MsfVenom,

– TaintDroid.

Построим матрицу характеристик каждой из альтернатив по выдвинутым критериям, которая представлена в таблице 2.1.

Таблица 2.1 – Характеристики средств проведения поведенческого анализа МПУ

	К1	К2	К3	К4	К5	К6	К7	К8	К9
TaintDroid	0	0	1	0	1	1	1	0	0,5
Crowdroid	1	1	0	1	1	1	1	1	0,5
MsfVenom	0	0	0	1	0	0	0,5	1	1

Определим важность каждого из критериев, для оценки эффективности средств проведения динамического анализа МПУ. Оценка критериев представлена в таблице 2.2. Показатель важности критерия – число от 1 до 9, где 1 – равная важность попарно сравниваемых критериев, а 9 – значимое превосходство i-го показателя на j-ым. Данная метрика называется шкалой относительной важности. После чего необходимо определить локальный вектор приоритетов исходя по формуле 2.1.

$$l_i = \frac{\sqrt[n]{\Pi_i}}{\sum \sqrt[n]{\Pi_i}}, \tag{2.1}$$

где l_i – i-ый элемент локального вектора приоритетов;

n – количество сравниваемых параметров;

a_{ij} – показатель превосходства i-го параметра над j-ым;

$П_i$ – произведение ряда показателей превосходства i-го параметра над остальными сравниваемыми.

Таблица 2.2 – Попарная оценка важности сравниваемых критериев

	К1	К2	К3	К4	К5	К6	К7	К8	К9	$\sqrt[з]{\prod_i a_{ij}}$	Локальный приоритет
К1	1,00	5,00	0,11	0,11	0,14	0,50	1,00	0,50	0,11	0,396997	0,026755
К2	0,20	1,00	0,11	0,11	0,14	0,33	5,00	0,50	0,11	0,317364	0,021388
К3	9,00	9,00	1,00	1,00	5,00	7,00	7,00	5,00	1,00	3,590708	0,241989
К4	9,00	9,00	1,00	1,00	5,00	7,00	7,00	5,00	1,00	3,590708	0,241989
К5	7,00	7,00	0,20	0,20	1,00	5,00	5,00	0,20	0,11	1,009507	0,068034
К6	2,00	3,00	0,14	0,14	0,20	1,00	0,50	0,33	0,11	0,425119	0,02865
К7	1,00	0,20	0,14	0,14	0,20	2,00	1,00	5,00	0,11	0,459154	0,030944
К8	2,00	2,00	0,20	0,20	5,00	3,00	0,20	1,00	0,11	0,72203	0,04866
К9	9,00	9,00	1,00	1,00	9,00	9,00	9,00	9,00	1,00	4,326749	0,291593

На текущем этапе анализа можно сделать вывод, что наиважнейшим критерием является корректность работы, а наименее важным критерием является простота использования средства.

После проведения попарного сравнения необходимо произвести проверку корректности оценки. Для этого вычислим индекс согласованности (ИС) согласно формуле 2.2.

$$ИС = \frac{\lambda_{max} - n}{n - 1}, \tag{2.2}$$

где λ_{max} – вычисляется по формуле 2.3;

n – количество сравниваемых параметров.

$$\lambda_{max} = \sum (E_j * l_i), \tag{2.3}$$

где E_j – сумма показателей превосходства j-го параметра над i-ым,

$l_i - j$-ый элемент ловакльного вектора приоритетов сравниваемых параметров.

После следует определить величину случайной согласованности (СС), исходя из размерности исследуемой матрицы. Значения этих величин представлены в таблице 2.3.

Таблица 2.3 – Величины случайной согласованности в зависимости от размерности изучаемой матрицы парных сравнений

Размерность матрицы	1	2	3	4	5	6	7	8	9	10
Величина случайной согласованности	0,00	0,00	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

Таким образом, для оценки важности критериев имеем:

$$\lambda_{max} = 10{,}14408$$

$$\text{ИС} = 0{,}14301$$

$$\text{СС} = 1{,}45$$

На данном этапе найдет отношение согласованности (ОС) по формуле 2.4.

$$\text{ОС} = \frac{\text{ИС}}{\text{СС}} = 0{,}098628 \qquad (2.4)$$

При показателе ОС не превышающем значение 0,1 можно утверждать, что суждения, на основании которых была построена матрица попарного сравнения не расходятся, а значит можно продолжить изучение средств проведения поведенческого анализа МПУ.

Произведем попарное сравнение анализируемых альтернатив аналогично приведенным ранее рассуждениям. В таблице 2.4 представлено сравнение средств поведенческого анализа МПУ по возможности проведения непрерывного анализа в режиме реального времени.

Таблица 2.4 – Определение приоритетов по критерию возможности проведения непрерывного анализа в режиме реального времени

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	0,33	3,00	1	0,258285
Crowdroid	3,00	1,00	5,00	2,466212	0,636986
MsfVenom	0,33	0,20	1,00	0,40548	0,104729
Сумма	4,33	1,53	9,00	3,87	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}038511$$

$$\text{ИС} = 0{,}019256$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}033199$$

Так как вычисленный параметр ОС не превышает значения 0,1, может сделать вывод, что превосходства сравниваемых параметров оценены корректно.

Определение приоритетов исследуемые средств проведения динамического анализа МПУ на возможность автоматизированного запуск представлено в таблице 2.5.

Таблица 2.5 – Парное сравнение по автоматизированный запуску

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	0,33	3,00	1	0,258285
Crowdroid	3,00	1,00	5,00	2,466212	0,636986
MsfVenom	0,33	0,20	1,00	0,40548	0,104729
Сумма	4,33	1,53	9,00	3,87	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}038511$$

$$\text{ИС} = 0{,}019256$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}033199$$

Следовательно, оценка проведена корректно.

Вычисление вектора приоритетов альтернатив по критерию

возможности анализа действий пользователя представлено в таблице 2.6.

Таблица 2.6 – Парное сравнение по анализу действий пользователя

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	3,00	3,00	2,080084	0,593634
Crowdroid	0,33	1,00	2,00	0,87358	0,249311
MsfVenom	0,33	0,50	1,00	0,550321	0,157056
Сумма	1,67	4,50	6,00	3,50	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}053622$$

$$\text{ИС} = 0{,}026811$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}046225$$

Следовательно, оценка проведена корректно.

Определим вектора приоритетов средств проведения поведенческого анализа функционирования МПУ на ОС андроид по критерию возможности анализа предположительно вредоносного ПО (таблица 2.7).

Таблица 2.7 – Определение приоритетов по возможности проведения анализа МПУ на наличие вредоносного ПО

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	0,20	0,20	0,341995	0,090909
Crowdroid	5,00	1,00	1,00	1,709976	0,454545
MsfVenom	5,00	1,00	1,00	1,709976	0,454545
Сумма	11,00	2,20	2,20	3,76	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3$$

$$\text{ИС} = 0$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 3$$

Следовательно, оценка проведена корректно.

В таблице 2.8 представлено сравнение средств поведенческого анализа МПУ по по фактору нейтральности сервиса для антивирусных систем.

Таблица 2.8 – Выявление приоритетов по фактору нейтральности сервиса для антивирусных систем

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	1,00	5,00	1,709976	0,497946
Crowdroid	1,00	1,00	2,00	1,259921	0,36689
MsfVenom	0,20	0,50	1,00	0,464159	0,135163
Сумма	2,20	2,50	8,00	3,43	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}094015$$

$$\text{ИС} = 0{,}047008$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}081048$$

Следовательно, оценка проведена корректно.

Оценим простоту использования альтернатив в таблице 2.9.

Таблица 2.9 – Парное сравнение по простоте использования

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	3,00	7,00	2,758924	0,65863
Crowdroid	0,33	1,00	4,00	1,100642	0,262753
MsfVenom	0,14	0,25	1,00	0,329317	0,078617
Сумма	1,48	4,25	12,00	4,19	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}032367$$

$$\text{ИС} = 0{,}016183$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}027902$$

Следовательно, оценка проведена корректно.

Вычисление вектора приоритетов альтернатив по критерию устойчивости к нестабильному соединению представлено в таблице 2.10.

Таблица 2.10 – Определение приоритетов по критерию устойчивости к нестабильному соединению

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	2,00	9,00	2,620741	0,582022
Crowdroid	0,50	1,00	9,00	1,650964	0,366651
MsfVenom	0,11	0,11	1,00	0,23112	0,051328
Сумма	1,61	3,11	19,00	4,50	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}053622$$

$$\text{ИС} = 0{,}026811$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}046225$$

Следовательно, оценка проведена корректно.

Вычисление вектора приоритетов альтернатив по доступности представлено в таблице 2.11.

Таблица 2.11 – Выявление приоритетов по критерию доступности сервиса

	TaintDroid	Crowdroid	MsfVenom	П	Локальный приоритет
TaintDroid	1,00	0,17	0,11	0,264567	0,063571
Crowdroid	6,00	1,00	1,00	1,817121	0,436622
MsfVenom	9,00	1,00	1,00	2,080084	0,499807
Сумма	16,00	2,17	2,11	4,16	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3{,}018295$$

$$\text{ИС} = 0{,}009147$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0{,}015771$$

Следовательно, оценка проведена корректно.

Определим вектора приоритетов средств проведения поведенческого анализа функционирования МПУ на ОС андроид по критерию корректности работы (таблица 2.12).

Таблица 2.12 – Определение приоритетов по корректности работы

	TaintDroid	Crowdroid	MsfVenom	$\sqrt[3]{\prod_i a_{ij}}$	Локальный приоритет
TaintDroid	1,00	1,00	0,50	0,793701	0,25
Crowdroid	1,00	1,00	0,50	0,793701	0,25
MsfVenom	2,00	2,00	1,00	1,587401	0,5
Сумма	4,00	4,00	2,00	3,17	1,00

Проверим корректность оценки.

$$\lambda_{max} = 3$$

$$\text{ИС} = 0$$

$$\text{СС} = 0{,}58$$

$$\text{ОС} = 0$$

Следовательно, оценка проведена корректно.

Вычислив локальные вектора приоритетов по каждому из критериев, построим матрицу для вычисления глобального приоритета. Для составим матрицу из векторов локального приоритета и вычислим элементы вектора глобального приоритета по формуле 2.5.

$$g_i = \sum a_{ij} * k_j, \tag{2.5}$$

где g_i – значение элемента вектора локальных приоритетов, соответствующий i-ой альтернативе и j-ому критерию;

a_{ij} – показатель вектора приоритета по j-му критерию для i-ой альтернативы;

k_j – значение важности j-ого критерия.

Вычисление глобального вектора приоритетов представлено в таблице 2.13.

Таблица 2.13 – Вычисление глобального вектора приоритетов сравнения альтернатив

Критерий	Векторы приоритетов									Глобальный приоритет
	К1	К2	К3	К4	К5	К6	К7	К8	К9	
Важность критерия	0,03	0,02	0,24	0,24	0,07	0,03	0,03	0,05	0,29	
TaintDroid	0,26	0,26	0,59	0,09	0,50	0,66	0,58	0,06	0,25	0,324834
Crowdroid	0,64	0,64	0,25	0,45	0,37	0,26	0,37	0,44	0,25	0,33897
MsfVenom	0,10	0,10	0,16	0,45	0,14	0,08	0,05	0,50	0,50	0,336196

Таким образом, на основе проведения иерархического анализа средств поведенческого анализа мобильных персональных устройств на ОС андроид, наиболее эффективным из рассматриваемых альтернатив является – Crowdroid.

Однако данное средство имеет ряд крупных недостатков:

– Вероятна ошибочная кластеризация при малых объемах данных, что может привести к ложному отнесению легитимных действий в системе к действиям, нарушающим безопасность устройства;

– Система всегда создает кластер с отнесением некоторых действий в системе к нелегитимными, даже если на устройстве нет нарушения защиты информационной безопасности;

– Даже малейшее изменения в отслеживаемых действиях может привести к перестройке кластеров, что замедляет вынесения вердикта о легитимности анализируемых действий;

– Средство использует небезопасное протокол связи FTP для сохранения собранной информации;

– Средство не производит анализ действий самого пользователя, как аналитику звонков, сообщений, геолокации устройства, что может привести к намеренной или непреднамеренной утечки пользовательской и коммерческой информации.

Выводы второго раздела

Существует множество различных средств поведенческого анализа функционирования мобильного устройства на операционной системе андроид. Однако они должным образом не решают вопрос контроля действий пользователя устройства, что зачастую является основным источником утечки информации на предприятии.

Помимо прочего, проверка системы с целью контроля пользовательских действий, может выступать в роли вспомогательного для уже функционирующих средств анализа и предотвращения угроз ИБ мобильного устройства, например, для проверки работоспособности антивирусных систем в случае заражения устройства.

3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

Целью реализации собственного решения является разработка средства поведенческого анализа мобильного персонального устройства на ОС андроид, которое будет лишено грубых известных изъянов, как возможное влияние на анализ устройства сторонними средствами и незащищенная передача анализируемых данных.

Также, было решено разработать средство, которое совместит в себе как анализ ПО мобильного устройства, так и контроль действий пользователя с целью предотвращения несанкционированного распространения информации.

В данном функционале для мобильных устройств будут заинтересованы не только пользователи МПУ для контроля целостности своих данных, но и предприятия, желающие контролировать утечки коммерческой информации, а также предотвращения нарушения информационной безопасности мобильных устройств своих сотрудников.

3.1 Разработка архитектуры сервиса поведенческого анализа функционирования мобильных устройств на андроид

Для реализации сервиса используется клиент-серверная архитектура, представленная на рисунке 3.1.

Мобильное приложение состоит из нескольких основных модулей, каждый из которых выполняет свою роль для поведенческого анализа функциональности системы Android. Каждый из них взаимодействует с модулем логирования, для отслеживания событий работы приложения, в случае каких-либо сбоев.

Серверное приложение состоит из базы данных, слоя работы с БД и API.

БД необходима для сохранения результатов работы мобильного приложения, а также их обработки. База данных имеет простую схему в виду одной таблицы (рисунок 3.2), для нормализации хранения информации и упрощенного сохранения и чтения данных из нее, но может быть расширена

как дополнительными таблицами, так и виртуальными таблицами.

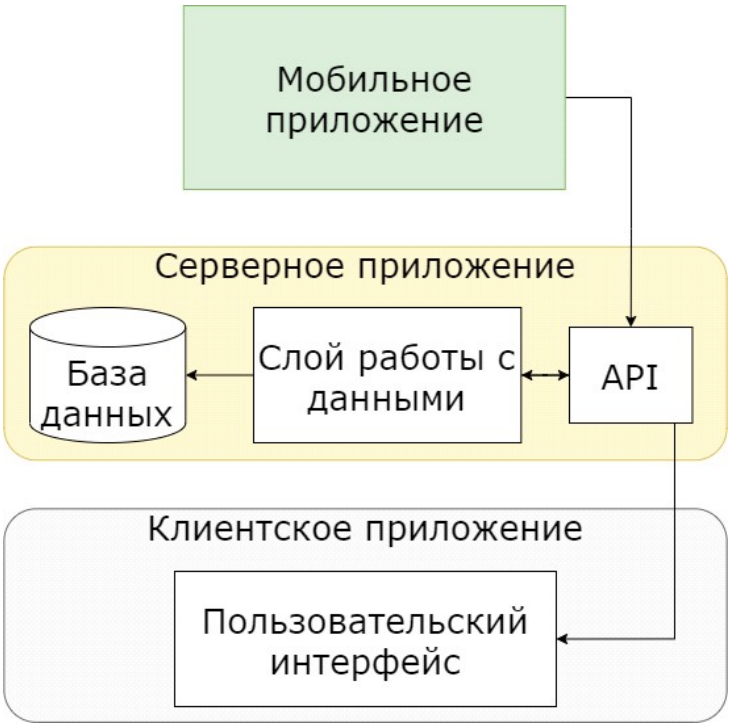

Рисунок 3.1 – Архитектура сервиса

Log			
PK	Id Guid NOT NULL		
	DeviceId	varchar(max)	NOT NULL
	Type	int	NOT NULL
	Created	date	NOT NULL
	Value	varchar(max)	NOT NULL

Рисунок 3.2 – Схема таблицы БД с данными

Слой работы с БД обеспечивает безопасное управление данными, хранящимися в базе данных, а также позволяет гибко настраивать ее и в любой момент времени обрабатывать актуальные данные.

API в свою очередь позволяет коммуницировать множеству функционирующих мобильных приложений и пользовательскому интерфейсу с системой, при этом с возможностью пред и постобработки данных.

3.2 Обоснование выбора инструментальных средств для разработки ПО

Для разработки данной системы были выбраны следующие технологии:

– Java – язык программирования, основное средство для реализации приложения для ОС на базе Android;

– .NET Core – платформа для разработки ПО [15], в рамках которого, с помощью языка программирования C# было реализовано серверное приложение данного сервиса;

– Entity Framework Core – объектно-ориентированная технология доступа к данным, является ORM решением для .NET Core платформы;

– React – JavaScript-библиотека для разработки пользовательских интерфейсов [16], с помощью которой на языке программирования TypeScript было написано клиентское приложение;

– IIS – набор сервисов для служб интернета [17], на которых функционирует серверное приложение.

Выбранный стек технологий является одним из наиболее популярных для решения подобных задач, так как данные средства имеют широкое сообщество, для решения большинства возникающих проблем в функционировании систем, реализованных на них, а помимо прочего обладают таким рядом преимуществ, как легкость и гибкость расширяемости, с целью увеличения производственных мощностей системы или расширения числа машин, взаимодействующих в системе.

В качестве сред разработки использовались:

– Android Studio – интегрированная среда разработки для работы с платформой Android [18],

– Microsoft Visual Studio 2019 – интегрированная среда разработки программного обеспечения и ряд других инструментальных средств [19],

– Microsoft Visual Studio Code – редактор исходного кода, разработанный Microsoft [20].

Разработка системы, и ее тестирование производилось на устройствах, чьи характеристики представлены в таблице 3.1.

Таблица 3.1 – Характеристики устройств разработки и тестирования

Приложение	Параметр	Разработка	Тестирование
Серверное	Устройство	Компьютер	Компьютер
	Процессор	Intel Core i5-3427U	Intel Core i5-3427U
	ПЗУ	100 Гб SSD	100 Гб SSD
	ОЗУ	8 Гб DDR3	8 Гб DDR3
	ОС	Windows 10 Pro x64	Windows 10 Pro x64
Клиентское	Устройство	Компьютер	Компьютер
	Процессор	Intel Core i5-3427U	Intel Core i5-3427U
	ПЗУ	100 Гб SSD	100 Гб SSD
	ОЗУ	8 Гб DDR3	8 Гб DDR3
	ОС	Windows 10 Pro x64	Windows 10 Pro x64
Мобильное	Устройство	Компьютер	Смартфон
	Процессор	Intel Core i5-3427U	Qualcomm Snapdragon 845
	ПЗУ	100 Гб SSD	110 Гб
	ОЗУ	8 Гб DDR3	6 Гб LPDDR4X
	ОС	Windows 10 Pro x64	Android 10

3.3 Общая схема работы системы

Целью работы является сбор данных с мобильных устройств для проведения поведенческого анализа. По этой причине, ввиду особенности системы Android, состоящей из множества взаимофункционирующих приложений, необходимо собрать данные функционирующих приложений, с целью анализа их взаимодействия. В связи с чем работа системы состоит из следующих этапов:

– Запуск возобновляемого фонового процесса, работающего со всеми модулями мобильного приложения;

– Последовательный запуск каждого модуля для сбора данных конкретной направления функционирования системы андроид (звонки, сетевые подключения, уведомления);

– Передача данных с конечного мобильного устройства на сервер обработки;

– Обработка полученных данных;

– Передача обработанных результатов работы мобильного устройства на клиентское приложение.

Во время работы приложения такие данные, как телефонные звонки, SMS, уведомления и так далее, собираются и отправляются на сервер обработки, о чем свидетельствует уведомление в шторке уведомлений МПУ.

3.4 Возобновляемый фоновый процесс

Для запуска мобильного приложения в фоновом процессе в ОС Android необходимо учитывать версию операционной системы, на котором работает устройство, в разных версиях ОС используются вызовы с разными API.

Для возобновляемого процесса необходимо создать дополнительный процесс, настроив его на определенный интервал перезапуска передаваемого в API-метод сервиса. Начиная с Android версии 5.0 данный интервал не может быть меньше минуты, в случае указания меньшего периода, система намеренно увеличит интервал до минуты без сопутствующих ошибок и вмешательства программиста в исходный код.

В созданном сервисе необходимо обработать запуск процесса, генерирующий для ОС старше версии Android 8.0 постоянное уведомление о работе сервиса. В противном случае приложение может быть заблокировано модераторами сервиса Google Play, в рамках которого распространяются приложения для устройств на операционных системах Android.

Также для корректной работы всех модулей необходим доступный из вне основного потока контекст сервиса, который используется ими для получения системных сервисов телефонии, смс-сообщений, геолокации и прочих.

3.5 Запуск модулей сбора данных

Для работы модулей сбора данных приложению необходимо доверенные разрешения к определенным физическим устройствам или системным сервисам.

Для корректной работы каждого модуля, при запуске приложения, проверяются все запрашиваемые разрешения, описанные в файле AndroidManifest.xml и вызывается повторный запрос разрешения в случае, если какое-то из разрешений не является доверенным.

В процессе, созданном в пункте 2.3, последовательно запускаются модули. Каждый из них может обладать вспомогательными функциями, которые позволят более гибко изменять логику работы модуля в связи с версионностью операционной системы, на которой работает приложение, так как было отмечено ранее, разные версии ОС обладают разными API для вызова схожих функций системы.

3.6 Основные модули сбора данных

Большая часть разработанных модулей работает по схоже принципу, однако для корректной работы некоторых требуется иной подход.

3.6.1 Модуль приложений

Модуль приложений обращается к пакетному менеджеру системы и запрашивает у него список установленных приложений. Данный модуль, помимо прочего, может получить данные и о предустановленных приложениях, которые могут быть скрыты от пользователя устройства. Далее составляется JSON-объект, содержащий в себе массив данных о полученных приложений, с указанием полного имени приложения, полном наименовании пакета приложения, а также информацию о версии этого приложения, что может быть полезно с точки зрения выявления возможных уязвимостей в данной версии обнаруженного приложения.

3.6.2 Модуль вызовов

Модуль вызовов собирает информацию о произведенных вызовах, содержащую:

– телефонный номер собеседника;

– наименование контакта, если данный номер сохранен в телефонной книге устройства;

– продолжительность звонка;

– дату звонка;

– тип звонка (исходящий, входящий, отклоненный и тп).

Данная информация позволит выявить возможного инсайдера, или предостеречь пользователя от возможного воздействия на него с помощью социальной инженерии.

3.6.3 Модуль контактов

Модуль контактов обращается к системному сервису контактов и с его помощью позволяет собрать информацию о номерах телефонов, сохраненных в телефонной книге, что также может быть полезным для обнаружения инсайдера.

3.6.4 Модуль геолокации

Модуль геолокации после проверки доступности предоставленного системного сервиса геолокации производит проверку возможности извлечения локации устройства по средствам GPS или интернета. После чего производит обнаружение координат устройства с помощью соответствующего системного GPS или сетевого сервиса, а также предоставляет информацию о точности найденной локации и скорости перемещения устройства. Приоритетным средством получения локации является модуль GPS, так как по средствам сети геопозиция может быть неточной при использовании прокси сервера или VPN тунелирования. Данный модуль полезен для обнаружения подозрительного перемещения устройства или его кражи.

3.6.5 Модуль SMS

Модуль смс, как и большинство модулей обращается к системному сервису, с помощью которого можно получить текст сообщения, статус его прочтения, тип сообщения (входящее, исходящее) и, в случае с входящим сообщением, ответное смс на него. Данный модуль так же, как модуль вызовов эффективен для обнаружения возможного инсайдера или возможных действий социальной инженерии.

3.6.6 Модуль Wi-Fi

Модуль Wi-Fi обращается к системному сервису контроля Wi-Fi подключения и собирает SSID и BSSID каждой Wi-Fi-сети, обнаруженной в данный момент устройством, что, как и модуль геопозиции, позволит определить расположение устройства в момент сбора информации.

3.6.7 Модуль уведомлений

В отличии от остальных модулей, модуль уведомлений функционирует не при запуске возобновляемого процесса, а при получении нового уведомления в системе. В связи с этим модуль унаследован от системного интерфейса слушателя, и настроен на просмотр текущих уведомлений, в связи с чем предоставление разрешений к данной функциональности системы отличается от остальных и требует разрешение на просмотр уведомлений устройства.

3.6.8 Модуль разрешений

Модуль разрешений позволяет получить информацию о доверенных и отвергнутых разрешениях, необходимых данному устройству, с целью проверки устройства на корректность работы всех модулей.

3.6.9 Модуль файлов

Данный модуль позволяет собрать информацию о состоянии файловой системы МПУ. Он предоставляет набор данных, содержащий информацию об имени исследуемого каталога, занимаемом им размере, а также аналогичную информацию о всех вложенных в него каталогов и файлах.

Статистика изменения размера каждого файла позволит проанализировать и обнаружить возможное нарушение ИБ.

Листинг всех разработанных модулей представлен в приложении А.

3.7 Передача данных на сервер

Для целостности собираемых данных при работе каждого из модулей мобильного приложения происходит передача собранной информации на сервер, по средства HTTP-запросов с использованием протокола SSL.

В случае неудачи передачи данных, обычно связанную с отсутствием сетевого подключения, следует сохранить собранную информацию и произвести передачу позднее, когда сетевое соединение будет возобновлено или иная проблема будет устранена.

3.8 Обработка полученных данных

Целью обработки полученных данных является трансформация данных для последующей их обработки, а также оптимизация и нормализация для сохранения полученной информации в БД.

Для этого принимающий сервер должен иметь более простой интерфейс данных, для устранения потери данных, но более сложный и полный для их хранения.

Выбранный стек технологий позволяет без труда внедрить данную систему в инфраструктуру с уже имеющейся базой данных, для чего необходимо изменить тип проводника данных в серверном приложении на необходимый.

3.9 Передача данных на клиентское приложение

Клиентское приложение при обращении к серверу получает набор данных, по которым можно определить устройство, над которым

производится анализ и сопутствующую информацию об активности устройства, что позволяет пользователю обработать эти под его конкретные нужды, как например выявление подозрительной активности голосовых и текстовых средств связи, подозрительных перемещениях мобильного устройства и прочих факторах, помогающих произвести дальнейший анализ собранной информации.

Клиентское приложение построено по принципу отрисовки со стороны сервера. Данный подход позволяет изменять содержимое отображаемой страницы без необходимости ее перезагрузки и позволяет избежать высоких затрат ресурсов устройства на отрисовку сайта со стороны пользователя, что является важным, а иногда критическим фактором для комфортного взаимодействия пользователя с веб-приложениями.

На каждом этапе функционирования системы производится запись ее функционирования с целью обнаружения возможных ошибок в ней и их устранения.

3.10 Использование системы поведенческого анализа мобильных устройств

Для использования разработанной системы необходим запуск серверного приложения с предварительной корректировкой настроек БД.

После чего необходимо сделать сервер доступным для внешней сети для корректной работы с удаленными мобильными устройствами.

Далее нужно установить на целевое мобильное устройство приложение, предоставив ему все запрашиваемые разрешения.

По завершении вышеописанных действия система начнет функционировать и процесс корректной работы данной системы можно будет наблюдать с помощью клиентского приложения.

Общий алгоритм работы системы, запускаемый каждую минуту, представлен на рисунке 3.3.

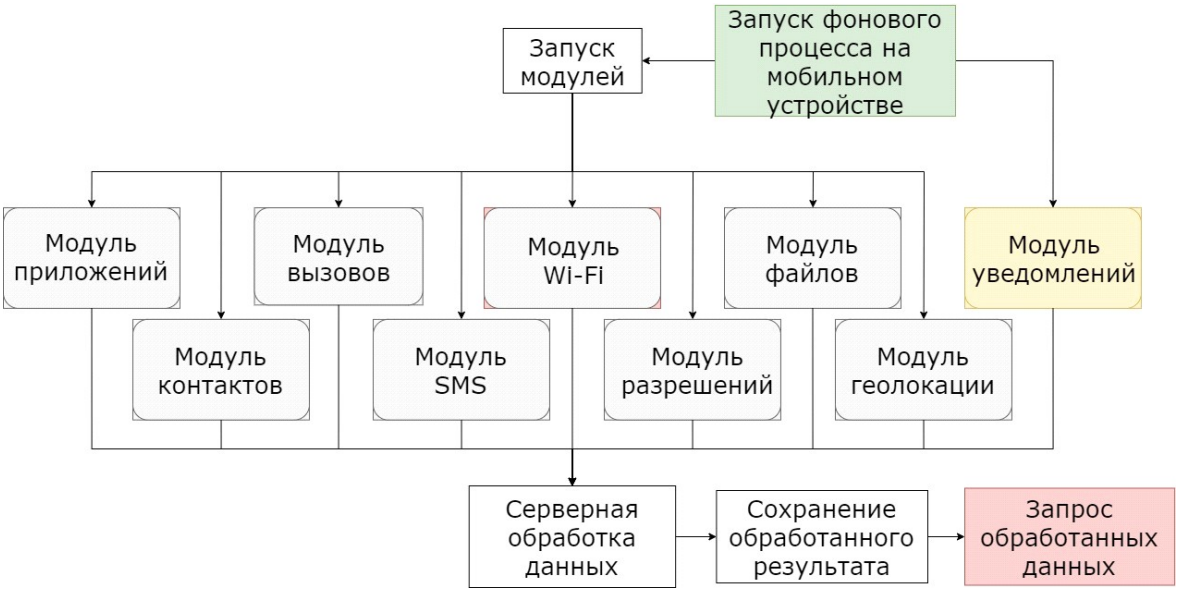

Рисунок 3.3 – Общий алгоритм работы системы

Как было отмечены выше и отражено на схеме алгоритма, модуль уведомлений работает по отличному от всех остальных принципу, что позволяет собирать информацию о всех пришедших уведомлениях без привязки к перезапуску процесса сбора информации. Так как в момент процесса сбора данных уведомления могут уже быть удалены пользователем или системой, именно такой подход позволит обеспечивать корректную работу данного модуля.

3.11 Примеры работы сервиса поведенческого анализа МПУ на ОС андроид

Как было отмечено ранее, основной аудиторией использования данного средства являются обеспечение целостности коммерческой информации. В связи с этим мобильное приложение, входящее в состав разработанной системы проведения поведенческого анализа МПУ, не влияет на процесс использования пользователем своего МПУ, так как не требует запуска в полноэкранном режиме и избавляет его от необходимости ручного запуска сервиса.

Но для информирования пользователя об активной работоспособности

сервиса, данное приложение создает постоянное уведомление о состоянии сервиса, пример представлен на рисунке 3.4.

Помимо этого, процесс работы сопровождается сообщениями от каждого работающего модуля, пример сообщения о работе сервиса, пример такого сообщения от модуля контактов продемонстрирован на рисунке 3.5, от модуля геолокации показано на рисунке 3.6, а от модуля Wi-Fi – на рисунке 3.7.

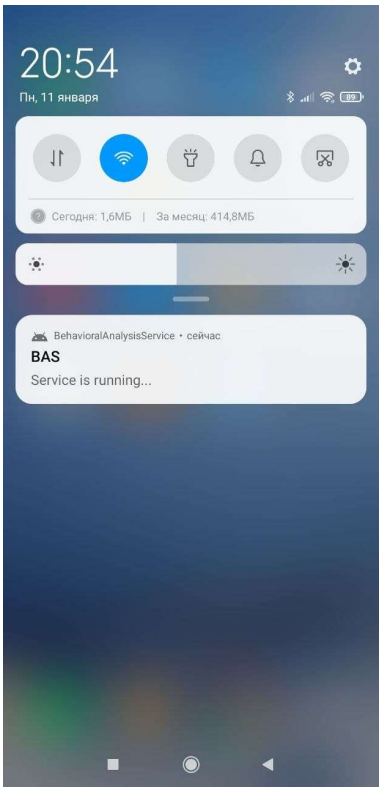

Рисунок 3.4 – Пример постоянного о работоспособности сервиса поведенческого анализа МПУ

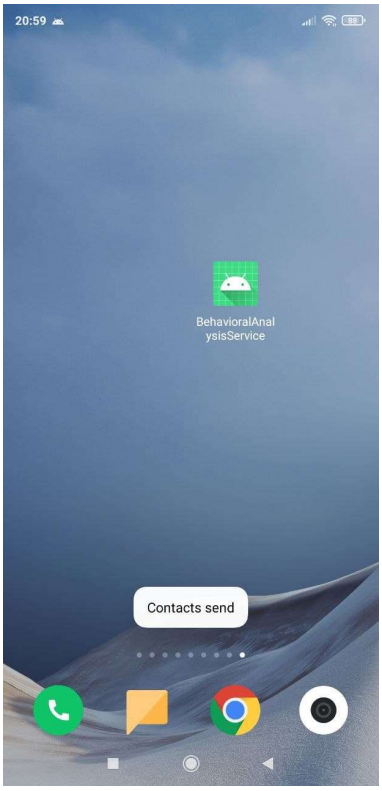

Рисунок 3.5 – Сообщение от модуля контактов

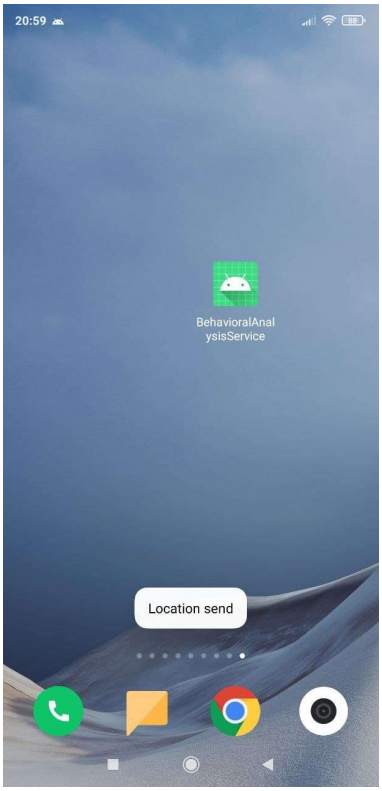

Рисунок 3.6 – Сообщение от модуля геолокации

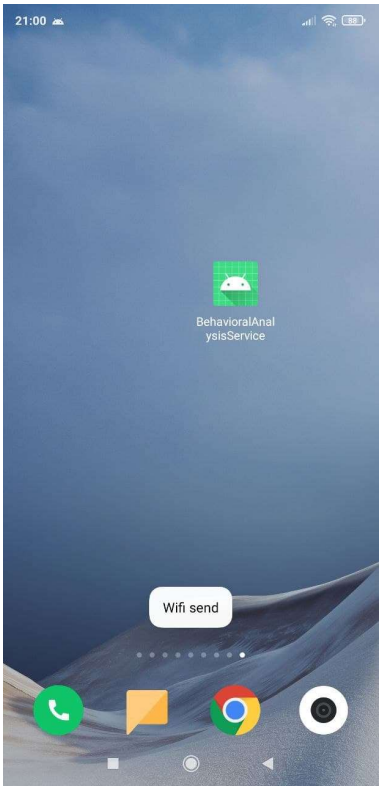

Рисунок 3.75 – Сообщение от модуля Wi-Fi

Все сообщения стандартизированы для четкого понимания о работе какого модуля пришло сообщение.

Сам интерфейс приложения упрощен, и обладает только индикатором функционирования системы, что освобождает пользователя от настройки работы приложения и является полезным для корпорации, желающей вести мониторинг действий своего сотрудника, без риска внесения изменений в общий принцип работы сервиса поведенческого анализа МПУ. Интерфейс приложения для андроид продемонстрирован на рисунке 3.8.

Для анализа собранной с устройства информации необходимо пользоваться клиентским веб-приложением, с помощью которого можно ознакомиться не только с собранными данными с устройства, представленные в формате JSON-строки, который пользуется наибольшей популярность в наше время для представления информации, но и с сопутствующей информацией как идентификатор МПУ, дата сбора данных и тип данных.

Рисунок 3.8 – Интерфейс приложения поведенческого анлиза МПУ для андроид

На рисунке 3.9 представлен интерфейс главной страницы клиентского приложения.

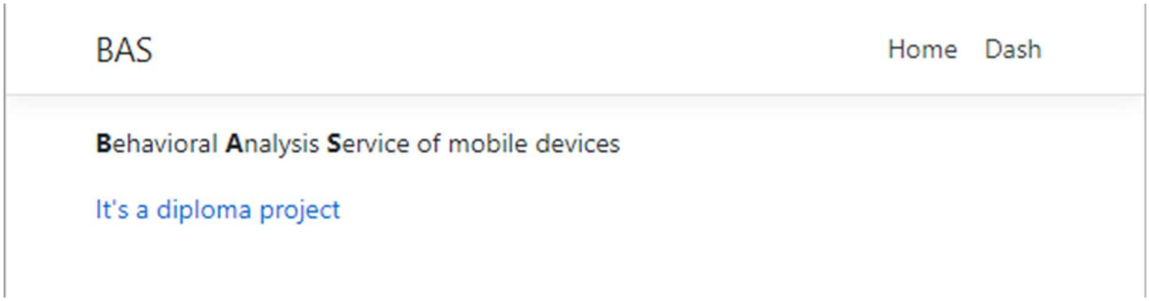

Рисунок 3.9 – Интерфейс главной страницы клиентского приложения

Главная страница содержит общую информацию о сервисе и кнопки для перехода на панель сервиса мониторинга.

На рисунке 3.10 представлен интерфейс клиентского приложения, позволяющий произвести дополнительный анализ собранных с МПУ данных.

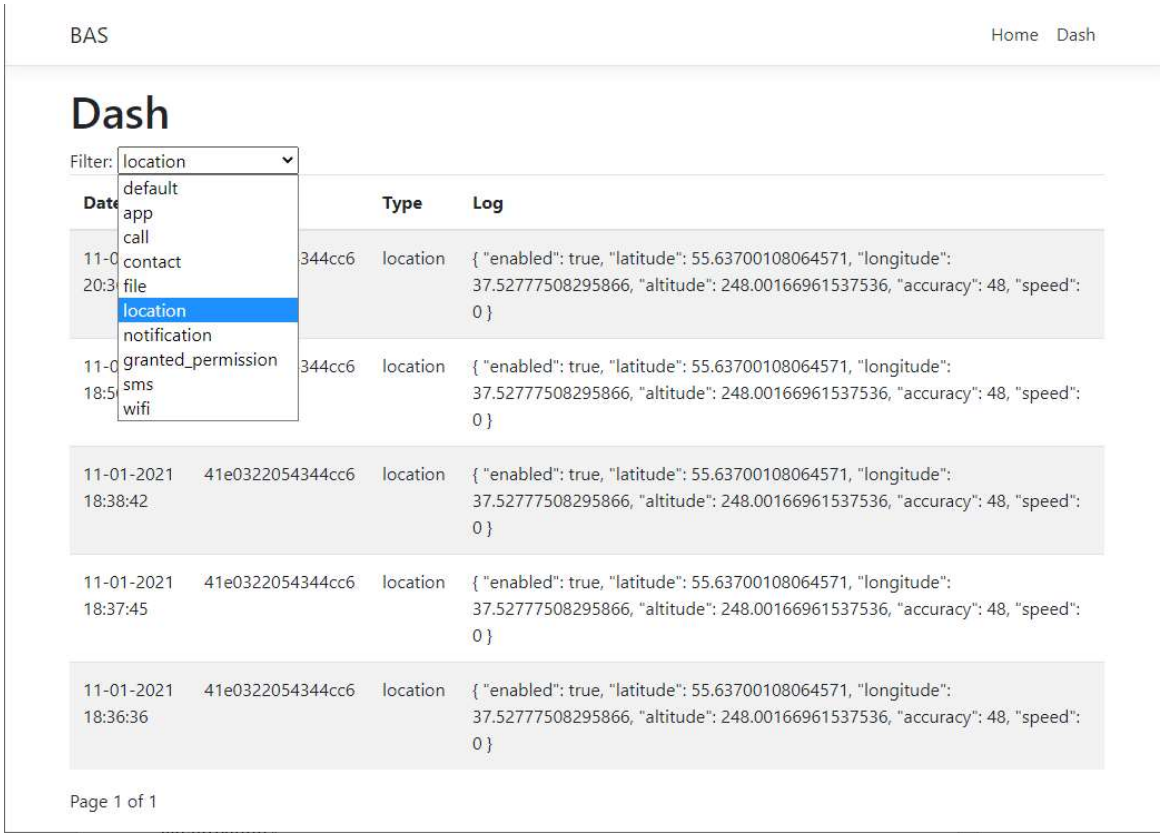

Рисунок 3.10 – Интерфейс страницы панели мониторинга

Как можно заметить, данный интерфейс обладает возможностью фильтрации данных по типу, а также имеет пагинацию, для упрощения взаимодействия с интерфейсом и ускоряет коммуникацию между клиентским и серверным приложением.

Выводы третьего раздела

Ввиду выбранного стека технологий данная система беспроблемно может быть масштабирована как по вертикали, так и по горизонтали, для чего не нужно производить серьезные правки существующего исходного кода, а достаточно реализовать звено, отвечающее за балансировку нагрузки.

Помимо прочего все уровни и модули системы реализованы наиболее популярными на текущий момент технологиями, что позволит легко и гибко изменять ее или дополнять новым функционалом специалисту, не обладающему высокими профессиональными навыками.

На ряду с аналогами, данная система обладает следующими достоинствами:

– Простота реализации, что является очень важным критерием для последующего поддержания и сопровождении системы;

– Все уровни и модули системы реализованы наиболее популярными на текущий момент технологиями, что позволит легко и гибко изменять ее или дополнять новым функционалом специалисту, не обладающему высокими профессиональными навыками;

– Модульность каждой части приложения, что обеспечивает слабосвязность каждого компонента системы и позволит безопасно для целостности системы модернизировать существующие или создавать новые;

– Легкость развертывания системы как для ограниченных, так и для крупно промышленных целей;

– Выбранный стек технологий позволит при необходимости масштабировать данную систему не только по вертикали, так и по горизонтали, не требуя для этого вмешательства в исходный код;

– Возможность контейнеризации системы таким средствами, как Docker и подобными.

4 ЭКОНОМИЧЕСКАЯ ОЦЕНКА ПРОЕКТА

Целью данного раздела является планирование и учет всевозможных затрат на разработку программного обеспечения, которые понесет компания при реализации системы поведенческого анализа мобильных устройств на операционной системе Android, ее внедрению, а также реализации средств администрирования данной системы. Будет проведена оценка следующих факторов: Затраты на разработку, расчет основных технико-экономических показателей.

4.1 Определение списка работ и состава исполнителей

Процесс разработки включает: обзор и анализ ПО со схожим функционалом, анализ и выбор программных продуктов для создания программы, отладку и тестирование. Каждый из этих этапов можно разделить на стадии, которые мы и будем оценивать.

В состав исполнителей, задействованных в разработке программного обеспечения для информационного поиска и классификации текстов, входят инженер-программист и руководитель проекта.

В таблице 4.1, приведенной ниже, содержится состав исполнителей и полный перечень работ.

Таблица 4.1 – Перечень работ и состав исполнителей

Стадии разработки	Наименование работы	Ответственные исполнители
1. Техническое задание	1.1. Постановка задачи	Руководитель проекта, инженер-программист
	1.2. Подбор и анализ технической литературы	Инженер-программист
	1.3. Сбор и анализ исходных данных	Инженер-программист
	1.4. Анализ существующих решений	Инженер-программист
	1.5. Определение этапов, сроков разработки системы и технической документации	Руководитель проекта, инженер-программист
	1.6. Согласование и утверждение технического задания	Руководитель проекта, инженер-программист

Продолжение таблица 4.1

Стадии разработки	Наименование работы	Ответственные исполнители
2. Эскизный проект	2.1. Подготовительные работы	Инженер-программист
	2.2. Выбор критериев эффективности и качества системы	Инженер-программист
	2.3. Разработка технологии тестирования и отладки	Инженер-программист
	2.4. Согласование и утверждение эскизного проекта	Руководитель проекта
3. Технический проект	3.1. Разработка технического задания	Руководитель проекта
	3.2. Разработка алгоритмического и программного обеспечения	Инженер-программист
	3.3. Разработка интерфейса	Инженер-программист
	3.4. Согласование и утверждение технического проекта	Руководитель проекта
4. Рабочий проект	4.1. Разработка программного решения	Инженер-программист
	4.2. Тестирование программного решения	Инженер-программист
	4.3. Отладка	Руководитель проекта
	4.4. Формирование программной документации	Руководитель проекта
	4.5. Согласование и утверждение рабочего проекта	Руководитель проекта
5. Внедрение	5.1. Подготовка и передача системы и технической документации для сдачи	Руководитель проекта
	5.2. Анализ данных, полученный в результате эксплуатации	Руководитель проекта
	5.3. Корректировки технической документации по результатам испытаний	Руководитель проекта

4.2 Насчет продолжительности и трудоемкости работ

Трудоемкость разработки определяется по сумме трудоемкости этапов и видов работ, оцениваемых экспертным путем в человеко-днях, и носит вероятностный характер, так как зависит от множества трудно учитываемых факторов.

Трудоемкость каждого вида работ определяется по формуле 4.1:

$$t_i = \frac{3 * t_{min} + 2 * t_{max}}{5}, \quad (4.1)$$

где t_i – трудоемкость работ, человек-дней;

t_{min}– минимально возможная трудоемкость выполнения отдельного вида работ в человеко-днях;

t_{max}– максимально возможная трудоемкость выполнения отдельного вида работ в человеко-днях.

Продолжительность каждого вида работ в календарных днях (Ti) определяется по формуле 4.2:

$$T_i = \frac{t_i}{Ч_i} * K_{вых}\ , \tag{4.2}$$

где t_i – трудоемкость работ, человек-дней;

$Ч_i$ – численность исполнителей, человек;

$K_{вых}$ – коэффициент, учитывающий выходные и праздничные дни, определяющийся по формуле 4.3:

$$K_{вых} = \frac{K_{кал}}{K_{раб}}, \tag{4.3}$$

где $K_{кал}$– число календарных дней;

$K_{раб}$– рабочие дни.

Для расчёта принимается среднее значение $K_{вых}$= 1,5.

Полный список видов и этапов работ по созданию программного продукта, экспертные оценки и расчетные величины их трудоемкости, а также продолжительность каждого вида работ, рассчитанные по приведенным выше формулам, представлены в таблице 4.2.

Таблица 4.2 – Расчёт трудоёмкости и продолжительности работ по созданию ПО.

№ работы	Наименование стадий и работ	Трудоемкость, чел. часы			Количество исполнителей, чел.	Продолжительность, календарные дни
		t_{min}	t_{max}	t_i	$Ч_i$	T_i
Технический проект						
1	Постановка задачи	2	4	2.8	2	2.1
2	Подбор литературы	1	2	1.4	1	2.1
3	Сбор и анализ исходных данных	1	2	1.4	1	2.1
4	Анализ существующих решений	2	4	2.8	1	4.2
5	Определение стадий, этапов и сроков разработки ПО	1	2	1.4	2	1.05
6	Согласование и утверждение технического задания	1	2	1.4	2	1.05
Эскизный проект						
8	Разработка структуры входных и выходных данных	1	2	1.4	1	2.1
9	Выбор критериев эффективности и качества системы	1	2	1.4	1	2.1
10	Разработка технологии тестирования и отладки	2	4	2.8	1	4.2
11	Согласование и утверждение эскизного проекта	1	2	1.4	1	2.1
Технический проект						
12	Разработка технического задания	2	4	2.8	1	4.2
13	Разработка алгоритмического и программного обеспечения	2	4	2.8	1	4.2
14	Разработка интерфейса	1	2	1.4	1	2.1

Продолжение таблица 4.2

№ работы	Наименование стадий и работ	Трудоемкость, чел. часы			Количество исполнителей, чел.	Продолжительность, календарные дни
		t_{min}	t_{max}	t_i	$Ч_i$	T_i
Рабочий проект						
15	Согласование и утверждение технического проекта	1	2	1.4	1	2.1
16	Разработка программного решения	15	20	17	1	25.5
17	Тестирование программного решения	4	6	4.8	1	7.2
18	Отладка	2	4	2.8	1	4.2
19	Разработка программной документации	2	6	3.6	1	5.4
20	Согласование и утверждение рабочего проекта	1	2	1.4	1	2.1
Внедрение						
21	Подготовка и передача системы и технической документации для сдачи	2	6	3.6	1	5.4
22	Анализ данных, полученных в результате эксплуатации	1	4	2.2	1	3.3
23	Корректировка технической документации по результатам испытаний	1	2	1.4	1	2.1
Итого				63.4		90.9

Судя по результатам расчетов, примерная трудоемкость проекта составит 63,4 человеко-часов, а продолжительность проекта — 90,9 календарных дней при условии последовательного выполнения работ.

4.3 Планирование разработки программного обеспечения с построением диаграммы Ганта

Используемая в качестве способа планирования диаграмма Ганта представляет собой графическое построение (диаграмму), на которой по вертикали приводится список выполняемых этапов и работ, а по горизонтальной шкале времени для каждой работы изображаются отрезки (прямоугольники), начало, конец и длина которых соответствуют началу, концу и длительности данной работы.

Подобные диаграммы применяются, когда этапы работы могут выполняться исключительно последовательно или в тех ситуациях, когда проект достаточно простой. В таких случаях диаграмма Ганта – подходящий инструмент планирования, так как она достаточно наглядная, простая и с её помощью можно легко представить и понять последовательность работ и действий, даты начала и окончания каждого этапа работы, и знать, что при соблюдении этих сроков разработка завершится в запланированные сроки.

Диаграмма Ганта разрабатываемого программного продукта, построенная по данным таблицы 4.2 приведена на рисунке 4.1.

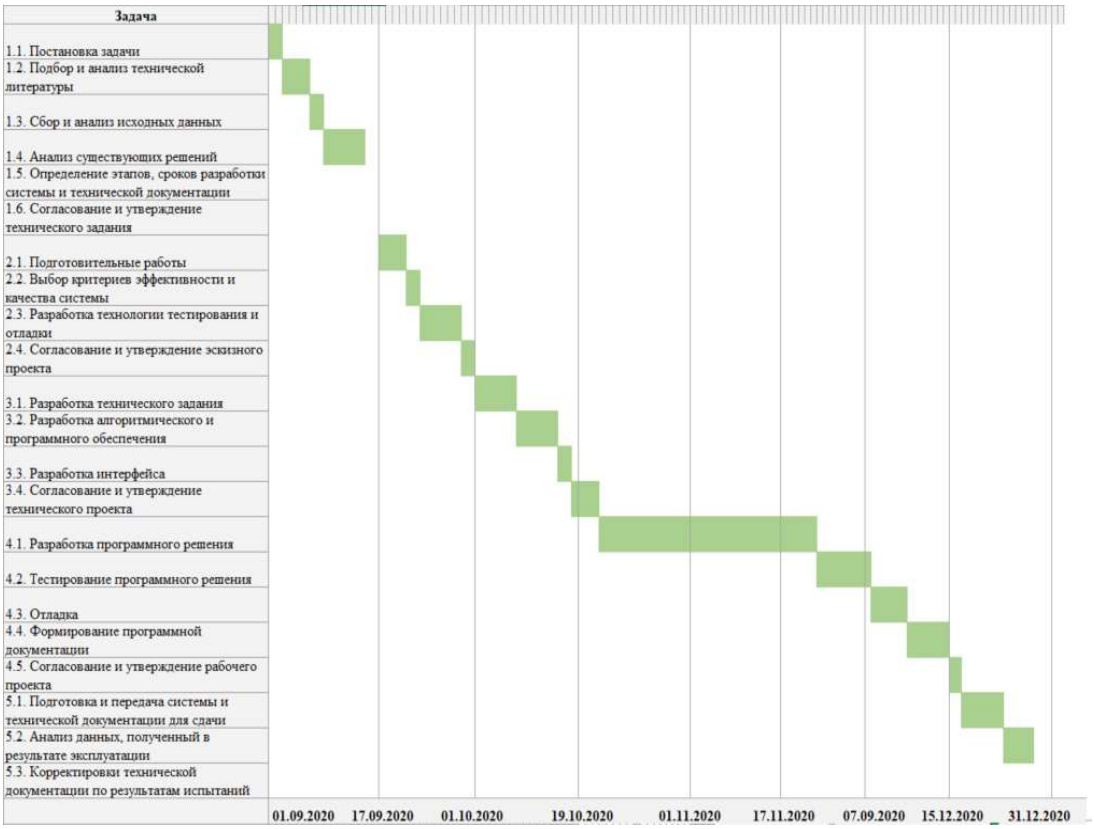

Рисунок 4.1 – Диаграмма Ганта

4.4 Расчет экономической эффективности

В выпускной квалификационной работе величину затрат на разработку ($C_{разр}$) программного продукта можно найти по формуле 4.4:

$$C_{разр} = C_{мат} + C_{осн} + C_{доп} + C_{м.вр} + C_{отч} + C_{накл} + C_{проч}, \quad (4.4)$$

где $C_{разр}$– стоимость разработки программного средства;

$C_{мат}$ – затраты на используемые материалы;

$C_{осн}$ – заработная плата исполнителей;

$C_{доп}$ – дополнительная заработная плата исполнителей (в среднем берется около 15% от Сосн);

$C_{м.вр}$ – затраты машинного времени;

$С_{отч}$ – отчисления от заработной платы исполнителей (пенсионный фонд, фонд обязательного медицинского страхования, фонд социального страхования, принимается, примерно, как 30% от $С_{осн}$ и $С_{доп}$;

$С_{накл}$ – накладные расходы (уборку, ремонт, и др., берется в размере около 60% от $С_{осн}$ и $С_{доп}$);

$С_{проч}$ – прочие раходы.

4.5 Расчет затрат на основные материалы

К этой статье расходов относят затраты на бумагу, канцтовары, комплектующие, дополнительное оборудование, а также включаем расходы на электроэнергию, которые можно рассчитать по формуле 4.5.

$$З_{эл} = P * Ц_{эл} * T_{и}, \quad (4.5)$$

где P – потребляемая мощность оборудования, кВт/ч;

$Ц_{эл}$– стоимость 1 кВт/ч, руб.;

$T_{и}$– время использования оборудования при проведении работ, ч.

$$З_{эл} = 0{,}072 * 5{,}47 * (90{,}9 * 8) = 286{,}4 \text{ руб.},$$

Эти затраты определяются экспертным путем. Пример расчета расходов показан в таблице 4.3.

Таблица 4.3 – Расчёт затрат на материалы.

№	Материалы, покупные изделия	Единица измерения	Цена, руб./ед.изм.	Количество, ед.изм.	Стоимость, руб.
1	Бумага А4	Пачка	235	1	235
2	Электроэнергия	кВт/час	5,47	52,35	286,4
3	Домен	Штука/год	300	1	300
4	Другие канцтовары	–	–	–	523,6
5	Итого, $С_{мат}$				1345

4.6 Заработная плата исполнителей

К статье «Заработная плата» относят оплату труда научных, инженерно-технических и других работников, которые принимают участие в разработке

проекта. Расчёт ведётся по формуле 4.6:

$$\mathrm{C_{осн}} = \mathrm{C_{ср}} * \mathrm{T}, \quad (4.6)$$

где $\mathrm{C_{осн}}$– заработная плата исполнителей (руб.);

T – трудоёмкость разработки (человеко-дни);

$\mathrm{C_{ср}}$ – средняя дневная тарифная ставка работника организации разработчика проекта(руб./человеко-дни), которая определяется по формуле 4.7:

$$C_{\mathrm{ср}} = \frac{C}{\Phi_{\mathrm{мес}}}, \quad (4.7)$$

где C – месячная зарплата работника (руб./мес.);

$\Phi_{\mathrm{мес}}$ – среднее количество рабочих дней в месяце (20 дней).

Дополнительная заработная плата на период разработки вычисляется по формуле 4.8:

$$\mathrm{C_{доп}} = \mathrm{C_{осн}} * 0{,}15 \quad (4.8)$$

Расчёт затрат на основную заработную плату разработчиков проекта приведен в таблице 4.4.

Таблица 4.4 – Затраты на заработную плату.

Исполнители	Месячная оплата труда (руб./мес.)	Дневная оплата труда (руб./дн.)	Время работы (рабоч.дн.)	Основная заработная плата(руб.)	Дополнительная заработная плата(руб.)	Общая величина заработной платы (руб.)
Руководитель проекта	63000	3000	34.05	102150	15322.5	117473
Инженер-программист	44625	2125	56.85	120806	18120.9	138927
Итого			90.9	222956	33443.4	256400

Кроме того, необходимо рассчитать, сколько потребуется внести средств в пенсионный и социальный фонды, а также в фонд медицинского страхования работника – то есть во все внебюджетные фонды.

Согласно формуле 4.9, затраты на внебюджетные фонды составляют:

$$С_{отч} = (С_{доп} + С_{осн}) * 0{,}30 = \ (33443{,}4 + 222956) * 0{,}3 = \\ = 76919{,}9 \text{ руб.} \quad (4.9)$$

4.6.1 Расчет накладных расходов

К статье «Накладные расходы» относят расходы, связанные с управлением и организацией работ, содержанием помещений (освещение, отопление, уборка и прочее). Накладные расходы рассчитывают относительно основной заработной платы, и их размер принимают равным 60% от основной заработной платы работников. Формула 4.10 расчёта представлена ниже:

$$С_{накл} = С_{осн} * К, \quad (4.10)$$

где $С_{накл}$– накладные расходы (руб.);

$С_{осн}$ – основная заработная плата исполнителей (руб.);

К – коэффициент учёта накладных расходов (К = 0,6).

Итак, в нашем случае:

$$С_{накл} = 222956 * 0{,}6 = 133773{,}8 \text{ руб.}$$

4.6.2 Расчёт стоимости машинного времени

Расчет стоимости машинного времени – это необходимые затраты на подготовку материалов, в том числе, документации, на разработку программного средства и иные расходы. Рассчитывается по формуле 4.11:

$$С_{м.вр} = К_{м.вр} * З_{м.вр}, \quad (4.11)$$

где $К_{м.вр}$ – стоимость одного часа машинного времени. Допустим, что в данной ситуации $К_{м.вр}$ = 3 руб/час;

$З_{м.вр}$ – машинное время, которое необходимо для создания проекта. Оно рассчитывается по формуле 4.12.

$$З_{м.вр} = t_i * Т_р * Т_{ск\ м.вр}, \quad (4.12)$$

где $Т_р$ – продолжительность рабочей смены ($Т_р = 8$ часов – длительность рабочего дня);

$Т_{ск\ м.вр}$ – это средний коэффициент использования машинного времени.

Большая часть проекта непосредственно связана с работой за

компьютером (работа с документацией в Word и разработка), поэтому $T_{\text{ск м.вр}}$ возьмем равной за 0,9.

Получим:

$$З_{\text{м.вр}} = 90{,}9 * 8 * 0{,}9 = \ 654{,}48 \text{ м. час}.$$

$$С_{\text{м.вр}} = 3 * 654{,}48 = 1963{,}44 \text{ руб.}$$

В результате расчетов можно вычислить итоговую себестоимость анализа проекта с учетом внедрения утилиты, которая рассчитывается по формуле 4.13. Рассчитаем себестоимость:

$$С_{\text{пр}} = 82080 \ + \ 47196 \ + 20520 + 136800 + 4731{,}57 = \\ = 291\ 327{,}573 \text{ руб.} \tag{4.13}$$

4.6.3 Общая смета затрат на разработку проекта

Результаты расчета затрат на разработку программного обеспечения для информационного поиска и классификации текстов представлены в таблице 4.5.

Таблица 4.5 – Смета затрат на разработку проекта.

Статья затрат	Обозначение	Величина затрат (руб.)	% затрат к итогу
Затраты на основные материалы	$С_{\text{мат}}$	1345.00	0.29%
Основная заработная плата	$С_{\text{осн}}$	222956.00	47.40%
Дополнительная заработная плата	$С_{\text{доп}}$	33443.40	7.11%
Отчисления от заработной платы	$С_{\text{отч}}$	76919.90	16.35%
Накладные расходы	$С_{\text{накл}}$	133773.80	28.44%
Машинное время	$С_{\text{м.вр}}$	1963.44	0.42%
Итого	$С_{\text{разр}}$	470401.54	100,00%

Как мы видим, себестоимость разработки составляет 470401,5 рубля.

4.7 Расчет окупаемости программного обеспечения

Данная программа может быть реализована на рынке. При расчётном количестве реализованных программ – n, оптовая цена программы ($Ц_{\text{опт}}$)

может быть рассчитана по формуле 4.14:

$$Ц_{опт} = \frac{С_{разр}}{n} + П_i , \quad (4.14)$$

где $Ц_{опт}$ – оптовая цена;

$С_{разр}$ – себестоимость разработки программы;

n – количество реализованных программ;

$П_i$ – прибыль, определяется по формуле 4.15:

$$П_i = У_р * \frac{С_{разр}}{n} , \quad (4.15)$$

где $У_р$ – средний уровень рентабельности, принимается $У_р$= 20%;

Таким образом, при среднем расчётном количестве реализованных программ n = 20 оптовая цена составит:

$$Ц_{опт} = 470401{,}5 / 20 + 0.2 * (470401{,}5 / 20) = 28224{,}1 \text{ руб.}$$

Отпускная оптовая цена реализации программы потребителям должна включать налог на добавленную стоимость (НДС) и рассчитывается по формуле 4.16:

$$Ц_{отп} = Ц_{опт} + НДС, \quad (4.16)$$

где НДС – налог на добавленную стоимость рассчитывается в соответствии с действующей ставкой этого налога: 20% от оптовой цены программы.

НДС = 5644,8 руб.

$Ц_{опт}$ = 28224,1 руб.

Таким образом, отпускная цена программы составит 28224,1руб., в том числе НДС – 5644,8 руб.

Затраты на внедрение системы ($С_{внедр}$) зависит только от затрат на разработку, так как система целиком и полностью будет эксплуатироваться на оборудовании заказчика: дополнительного оборудования для внедрения данной системы не требуется. Также не требуется трата средств на дополнительное программное обеспечение для внедрения.

$$С_{внедр} = С_{разр} = 470401{,}5 \text{ руб.}$$

Итак, затраты на сопровождение системы ($С_{сопр}$) зависят только от затрат

на оплату труда. Однако покупка пакета поддержки не является обязательной услугой.

$$C_{сопр} = C_{зарп} = 0 \text{ руб.}$$

Для расчета срока окупаемости систем используют формулу 4.17:

$$T = \frac{C_{внедр}}{0.8 * (Z - C_{сопр})}, \quad (4.17)$$

где T – срок окупаемости системы;

Z – ожидаемая прибыль (28224,1 руб./мес.).

$$T = \frac{470401,5}{0.8 * (28224,1 - 0)} = 20,9 \text{ мес.}$$

Выводы четвертого раздела

В этом разделе были определены и рассчитаны продолжительность и трудоемкость разработки, была построена диаграмма Ганта и произведен расчет окупаемости программного обеспечения, а также общих затрат.

Судя по расчёту экономической эффективности, большая часть затрат при разработке программного продукта будет приходиться на основную заработную плату (47,4%), а минимум от всех затрат (0,29%) придется на основные материалы, необходимые для разработки.

Также были произведены расчёты, согласно которым окупаемость (период времени, нужный для того, чтобы начать получать прибыль, вернув средства, затраченные на разработку программного обеспечения) равна 20.9 месяцев.

ЗАКЛЮЧЕНИЕ

В данной работе мной была изучена проблема обеспечения безопасности мобильных устройств, которые стали наиболее популярными видом устройств в мире за последние 5 лет. Был проанализирован темп роста популярности операционных систем для МПУ с целью выбора целевой ОС для последующего исследования.

Ввиду непрерывного роста популярности операционной системы андроид, было принято решение исследовать данную систему, чтобы понять принципы обеспечения безопасности мобильных устройств.

Для обеспечения целостности и конфиденциальности информации, хранящейся на мобильном устройстве, необходимо использовать сервисы поведенческого анализа функционирования МПУ, для обеспечения защиты и мониторинга ИБ в процессе его эксплуатации.

Наиболее заинтересованной стороной с точки зрения контроля утечек и нарушения целостности конфиденциальных данных являются компании для пресечения раскрытия коммерческих тайн и несанкционированного проникновения в информационную систему предприятия через личные мобильные устройства своих сотрудников. Для обеспечения такого рода потребности средство должно обладать не только функционалом анализа поведения операционной системы, но и контроля деятельности пользователя устройства.

Были проанализированы существующие средства, для проведения динамического анализа состояния информационной безопасности мобильных устройств. Были рассмотрены и выделены такие сильные стороны, как использование нейронных сетей и машинного обучения для автоматического определения легитимности производимых на устройстве действий. Однако был выдвинут ряд недостатков этих средств.

Для выделения наиболее эффективного средства из рассмотренных был проведен их анализ методом иерархий, по итогу которого было определено

средство для исследования его существенных недостатков.

Учитывая выделенные недостатки в рамках данной работы мной было разработано и реализовано собственное решение проведения поведенческого анализа мобильных устройств на операционной системе андроид в виде перезапускаемого сервиса. Разработанное решение содержит в себя ряд модулей, доступный для расширения и уже обеспечивающий возможность контроля как целостности системы мобильного устройства, так и действий его пользователя.

Реализованный сервис проведения поведенческого анализа функционирования мобильных устройств на операционной системе андроид не требует глубоких технических знаний для внедрения его как в корпоративных, так и личных целях.

При этом разработанная архитектура обладает рядом преимуществ, позволяющих легко и гибко дополнять функциональность реализованного сервиса как в приложение для ОС андроид, так и в клиент-серверную часть системы, благодаря использованию наиболее популярных средств реализации подобных информационных систем.

Также, коммуникация между блоками реализованной системы происходит по защищенному каналу передачи данных, в отличии от некоторых сравниваемых аналогов, передающих данные между своими узлами по несовременным или плохо защищенным каналам связи.

Таким образом предустановка разработанного в данной работе программного обеспечения контроля целостности информации предполагается в основном на корпоративные устройства, используемые в рабочих целях, с согласия держателя мобильного персонального устройства.

Была проанализирована экономическая эффективность реализованного сервиса, согласно которой окупаемость затрат на производство подобного ПО составит 21 месяц, после чего начнет приносить прибыль в размере 28244,1 рубля в месяц.

СПИСОК ИСТОЧНИКОВ

1) [Электронный ресурс]: Statcounter GlobalStats. – URL: https://gs.statcounter.com/ (дата обращения 21.10.2020)

2) Монаппа К. А. Анализ вредоносных программ / пер. с англ. Беликова Д. А. – М.: ДМК Пресс, 2019. – 452с.

3) [Электронный ресурс]: Positive Technologies. – URL: https://www.ptsecurity.com/ru-ru/research/analytics/financial-application-vulnerabilities-2018/ (дата обращения 15.10.2020)

4) [Электронный ресурс]: Smart Security. URL: https://bosfera.ru/pdf/16.%20%D0%A0%D0%B8%D0%BD%D0%B0%D1%82%20%D0%90%D0%BD%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D0%B2.pdf (дата обращения 18.10.2020)

5) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Android (дата обращения 21.10.2020)

6) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Windows (дата обращения 21.10.2020)

7) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/IOS (дата обращения 21.10.2020)

8) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Windows_10_Mobile (дата обращения 21.10.2020)

9) [Электронный ресурс]: Anti-Malware. – URL: https://www.anti-malware.ru/analytics/Threats_Analysis/principles_detection_malicious_applications_Android (дата обращения 22.10.2020)

10) T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. 10-я конференция Annual Network & Distributed System Security Symposium (NDSS), 2003

11) W. Enck, P. Gilbert, B.-G. Chunn, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime

Privacy Monitoring on Smartphones. 9-я конференция USENIX. Симпозиум по дизайну и применению операционных систем 2010

12) G. Sarwar, O. Mehani, R. Boreli, and M. A. Kaafar. On the Effectiveness of Dynamic Taint Analysis for Protecting Against Private Information Leaks on Androidbased Devices. 10-я международная конференция International Conference on Security and Cryptography (SECRYPT), 2013

13) Y. Zhang, M. Yang, B. Xu, Z. Yang, G. Gu, P. Ning, X. Wang, B. Zang. Vetting Undesirable Behaviors in Android Apps with Permission Use Analysis. 20-я конференцияACM Conference on Computer and Communications Security (CCS), 2013

14) I. Burguera, U. Zurutuza, S. Nadjm-Tehrani. Crowdroid: Behavior-Based Malware Detection System for Android. 1-я конференция ACM workshop on Security and privacy in smartphones and mobile devices, 2011

15) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/.NET_Core (дата обращения 10.01.2021)

16) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/React (дата обращения 10.01.2021)

17) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Internet_Information_Services (дата обращения 10.01.2021)

18) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Android_Studio (дата обращения 10.01.2021)

19) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2019 (дата обращения 10.01.2021)

20) [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Visual_Studio_Code (дата обращения 10.01.2021)

ПРИЛОЖЕНИЕ А

Возобновляемый сервис

```
package com.behavioralanalysis.service;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import java.util.ArrayList;
import java.util.List;
public class MainActivity extends Activity {
    public static final String CHANNEL_ID = "exampleServiceChannel";
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        createNotificationChannel();
        setContentView(R.layout.activity_main);
        requestNotificationPermissionIfNeed();
        requestPermissionIfNeed();
        if (checkNotificationService() && getNotGrantedPermissions().isEmpty()) {
            Intent serviceIntent = new Intent(this, MainService.class);
            PendingIntent pendingIntent = null;
            if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O)
{
                pendingIntent = PendingIntent.getForegroundService(this, 1,
serviceIntent, 0);
            } else {
                pendingIntent = PendingIntent.getService(this, 1, serviceIntent, 0);
            }
            AlarmManager alarmManager =
(AlarmManager)getSystemService(ALARM_SERVICE);
            alarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP, 0, 10000,
pendingIntent);
        } else {
            finish();
        }
    }
    private boolean checkNotificationService() {
        ContentResolver contentResolver = getContentResolver();
        String enabledNotificationListeners =
                Settings.Secure.getString(contentResolver,
"enabled_notification_listeners");
        String packageName = getPackageName();
        return enabledNotificationListeners != null &&
enabledNotificationListeners.contains(packageName);
    }
```

```
    private void requestNotificationPermissionIfNeed() {
        if (!checkNotificationService()) {
            startActivity(new
Intent(Settings.ACTION_NOTIFICATION_LISTENER_SETTINGS));
        }
    }
    private List<String> getNotGrantedPermissions() {
        PackageInfo packageInfo = null;
        String[] permissions = new String[]{};
        Context context = getApplicationContext();
        try {
            packageInfo =
getPackageManager().getPackageInfo(context.getPackageName(),
PackageManager.GET_PERMISSIONS);
            permissions = packageInfo.requestedPermissions;
        } catch (PackageManager.NameNotFoundException ex) {
            ex.printStackTrace();
        }
        List<String> neededPermissions = new ArrayList<>();
        for (String permission : permissions) {
            int status = ContextCompat.checkSelfPermission(this, permission);
            if (status != PackageManager.PERMISSION_GRANTED) {
                neededPermissions.add(permission);
            }
        }
        return neededPermissions;
    }
    private void requestPermissionIfNeed() {
        List<String> neededPermissions = getNotGrantedPermissions();
        if (!neededPermissions.isEmpty()) {
            Toast.makeText(
                    this,
                    "Для работы приложения необходимо предоставить разрешения",
                    Toast.LENGTH_LONG
            ).show();
            ActivityCompat.requestPermissions(
                    this,
                    neededPermissions.toArray(new String[neededPermissions.size()]),
                    0
            );
        }
    }
    private void createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel serviceChannel = new NotificationChannel(
                    CHANNEL_ID,
                    "Example Service Channel",
                    NotificationManager.IMPORTANCE_DEFAULT
            );
            NotificationManager manager =
getSystemService(NotificationManager.class);
            manager.createNotificationChannel(serviceChannel);
        }
    }
}
package com.behavioralanalysis.service;
import android.app.Notification;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
```

```
import android.os.IBinder;
import android.provider.Settings;
import androidx.annotation.Nullable;
import androidx.core.app.NotificationCompat;
public class MainService extends Service {
    private static Context context;
    public static String id = "";
    @Override
    public void onCreate() {
        super.onCreate();
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Intent notificationIntent = new Intent(this, MainActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
notificationIntent, 0);
        Notification notification = new NotificationCompat.Builder(this,
MainActivity.CHANNEL_ID)
                .setContentTitle("BAS")
                .setContentText("Service is running...")
                .setSmallIcon(R.drawable.ic_android)
                .setContentIntent(pendingIntent)
                .build();
        startForeground(1, notification);
        context = this;
        id = Settings.Secure.getString(getContext().getContentResolver(),
                Settings.Secure.ANDROID_ID);
        Sender.start();
        return START_NOT_STICKY;
    }
    @Override
    public void onDestroy() {
        Logger.log("MainService.onDestroy()");
        super.onDestroy();
        sendBroadcast(new Intent("respawnService"));
    }
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    public static Context getContext() {
        return context;
    }
}
package com.behavioralanalysis.service;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
public class ServiceReciever extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Intent _intent = new Intent(context, MainService.class);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            context.startForegroundService(_intent);
        } else {
            context.startService(_intent);
        }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.behavioralanalysis.service">
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <application
        android:requestLegacyExternalStorage="true"
        android:allowBackup="true"
        android:hardwareAccelerated="false"
        android:largeHeap="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:usesCleartextTraffic="true"
        android:theme="@style/AppTheme">
        <uses-library android:name="org.apache.http.legacy"
android:required="false"/>
        <activity
            android:name=".MainActivity"
            android:launchMode="singleInstance">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".MainService"
            android:enabled="true"
            android:exported="false"
            />
        <receiver
            android:name=".ServiceReciever"
            android:enabled="true"
            android:exported="true"
            android:label="RestartServiceWhenStopped">
            <intent-filter>
                <action android:name="respawnService" />
            </intent-filter>
        </receiver>
        <service
            android:name=".NotificationListener"
            android:label="@string/app_name"
```

```
android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE">
            <intent-filter>
                <action
android:name="android.service.notification.NotificationListenerService" />
            </intent-filter>
        </service>
    </application>
</manifest>
```

Модуль сбора данных

```
package com.behavioralanalysis.service;
import android.content.Context;
import android.os.AsyncTask;
import android.telecom.Call;
import android.widget.Toast;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Calendar;
import javax.net.ssl.HttpsURLConnection;
public class Sender extends AsyncTask<String, Void, String> {
    public static String baseUrl = "https://basdiploma.site/api/log/";
    @Override
    protected String doInBackground(String... strings) {
        //Logger.log("sending" + strings[0]);
        String data = "";
        URL url;
        try {
            url = new URL(baseUrl);
            HttpsURLConnection httpURLConnection =
(HttpsURLConnection)url.openConnection();
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setRequestProperty("Content-Type","application/json");
            httpURLConnection.setRequestProperty("Accept", "application/json");
            httpURLConnection.connect();
            OutputStream outputStream = httpURLConnection.getOutputStream();
            OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(outputStream, "UTF-8");
            outputStreamWriter.write(strings[0]);
            outputStreamWriter.flush();
            outputStreamWriter.close();
            BufferedReader bufferedReader = new BufferedReader(
                    new InputStreamReader(httpURLConnection.getInputStream(), "UTF-
8")
            );
            String line = null;
            StringBuilder sb = new StringBuilder();
            while ((line = bufferedReader.readLine()) != null) {
                sb.append(line);
            }
            bufferedReader.close();
            data = sb.toString();
        } catch (Exception ex) {
            Logger.log("error", ex.getMessage());
            ex.printStackTrace();
```

```
        }
        return data;
    }
    public static void start() {
        Logger.log("Sender.start()");
        try {
            Sender.sendApps(true);
            Sender.sendContacts();
            Sender.sendLocation();
            Sender.sendPermissions();
            Sender.sendSms();
            Sender.sendWifi();
            Sender.sendFile("/storage/emulated/0");
            Sender.sendCalls();
        } catch (Exception ex) {
            ex.printStackTrace();
            start();
        }
    }
    public static void send(JSONObject toSend) {
        Logger.log("Sender.send()");
        try {
            toSend.put(
                    "date",
                    android.text.format.DateFormat.format(
                            "yyyy-MM-dd HH:mm:ss",
                            Calendar.getInstance().getTime()
                    )
            );
            toSend.put("deviceId", MainService.id);
            new Sender().execute(toSend.toString());
        } catch (JSONException ex) {
            ex.printStackTrace();
        }
    }
    public static void showMsg(String msg) {
        Toast.makeText(MainService.getContext(), msg, Toast.LENGTH_SHORT).show();
    }
    public static boolean sendWifi() {
        Logger.log("Sender.sendWifi()");
        try {
            showMsg("Wifi send");
            JSONObject data = new JSONObject();
            data.put("type", "wifi");
            data.put("value", Wifi.get());
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Wifi failed");
            return false;
        }
        return true;
    }
    public static boolean sendSms() {
        Logger.log("Sender.sendSms()");
        try {
            showMsg("Sms send");
            JSONObject data = new JSONObject();
            data.put("type", "sms");
            data.put("value", Sms.get());
            send(data);
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Sms failed");
            return false;
        }
        return true;
    }
    public static boolean sendPermissions() {
        Logger.log("Sender.sendPermissions()");
        try {
            showMsg("Permissions send");
            JSONObject data = new JSONObject();
            data.put("type", "granted_permission");
            data.put("value", Permissions.getGranted());
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Permissions failed");
            return false;
        }
        return true;
    }
    public static boolean sendNotification(JSONObject notification) {
        Logger.log("Sender.sendNotification()");
        try {
            showMsg("Notification send");
            JSONObject data = new JSONObject();
            data.put("type", "notification");
            data.put("value", notification);
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Notification failed");
            return false;
        }
        return true;
    }
    public static boolean sendLocation() {
        Logger.log("Sender.sendLocation()");
        try {
            showMsg("Location send");
            Context context = MainService.getContext();
            Locations gps = new Locations(context);
            if (gps.isCanGetLocation()) {
                JSONObject data = new JSONObject();
                data.put("type", "location");
                data.put("value", gps.get());
                send(data);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Location failed");
            return false;
        }
        return true;
    }
    public static boolean sendContacts() {
        Logger.log("Sender.sendContacts()");
        try {
            showMsg("Contacts send");
            JSONObject data = new JSONObject();
```

```
            data.put("type", "contact");
            data.put("value", Contacts.get());
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Contacts failed");
            return false;
        }
        return true;
    }
    public static boolean sendCalls() {
        Logger.log("Sender.sendCalls()");
        try {
            showMsg("Calls send");
            JSONObject data = new JSONObject();
            data.put("type", "call");
            data.put("value", Calls.get());
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Calls failed");
            return false;
        }
        return true;
    }
    public static boolean sendApps(boolean isWithSystemApps) {
        Logger.log("Sender.sendApps()");
        try {
            showMsg("Apps send");
            JSONObject data = new JSONObject();
            data.put("type", "app");
            data.put("isWithSystemApps", isWithSystemApps);
            data.put("value", Apps.get(isWithSystemApps));
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Apps failed");
            return false;
        }
        return true;
    }
    public static boolean sendError(JSONObject data) {
        Logger.log("Sender.sendError()");
        try {
            data.put("type", "error");
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("Error failed");
            return false;
        }
        return true;
    }
    public static boolean sendFile(String path) {
        Logger.log("Sender.sendFile()");
        try {
            showMsg("Files send");
            JSONObject data = new JSONObject();
            data.put("type", "file");
            data.put("value", Files.get(path));
            send(data);
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("File failed");
            return false;
        }
        return true;
    }
    public static boolean downloadFile(JSONObject object) {
        Logger.log("Sender.sendFile()");
        try {
            JSONObject data = new JSONObject();
            data.put("type", "download_file");
            data.put("value", object);
            send(data);
        } catch (Exception ex) {
            ex.printStackTrace();
            showMsg("File failed");
            return false;
        }
        return true;
    }
}
```

Модуль приложений

```
package com.behavioralanalysis.service;
import android.content.Context;
import android.content.pm.PackageInfo;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.List;
public class Apps {
    public static JSONArray get(boolean withSystemPackages) {
        JSONArray array = new JSONArray();
        try {
            Context context = MainService.getContext();
            List<PackageInfo> packages = context
                    .getPackageManager()
                    .getInstalledPackages(0);
            for (int i = 0; i < packages.size(); i++) {
                PackageInfo packageInfo = packages.get(i);
                if (!withSystemPackages && packageInfo.versionName == null) {
                    continue;
                }
                try {
                    JSONObject object = new JSONObject();
                    object.put(
                            "appName",
                            packageInfo.applicationInfo.loadLabel(
                                    context.getPackageManager()
                            ).toString()
                    );
                    object.put("packageName", packageInfo.packageName);
                    object.put("varsionName", packageInfo.versionName);
                    object.put("versionCode", packageInfo.versionCode);
                    array.put(object);
                } catch (JSONException ex) {
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
```

```
        }
        return array;
    }
}
```

Модуль вызовов

```
package com.behavioralanalysis.service;
import android.database.Cursor;
import android.net.Uri;
import android.provider.CallLog;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
public class Calls {
    public static JSONArray get() {
        JSONArray array = new JSONArray();
        Uri allCalls = Uri.parse("content://call_log/calls");
        Cursor cursor = MainService.getContext().getContentResolver().query(allCalls,
null, null, null, null);
        try {
            while (cursor.moveToNext()) {
                JSONObject object = new JSONObject();
                object.put(
                        "number",
                        cursor.getString(
                                cursor.getColumnIndex(CallLog.Calls.NUMBER)
                        )
                );
                object.put(
                        "name",
                        cursor.getString(
                                cursor.getColumnIndex(CallLog.Calls.CACHED_NAME)
                        )
                );
                object.put(
                        "duration",
                        cursor.getString(
                                cursor.getColumnIndex(CallLog.Calls.DURATION)
                        )
                );
                object.put(
                        "date",
                        cursor.getString(
                                cursor.getColumnIndex(CallLog.Calls.DATE)
                        )
                );
                object.put(
                        "type",
                        Integer.parseInt(cursor.getString(
                                cursor.getColumnIndex(CallLog.Calls.TYPE)
                        ))
                );
                array.put(object);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return array;
    }
}
```

Модуль контактов

```
package com.behavioralanalysis.service;
import android.database.Cursor;
import android.provider.ContactsContract;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
public class Contacts {
    public static JSONArray get() {
        JSONArray array = new JSONArray();
        try {
            Cursor cursor = MainService
                    .getContext()
                    .getContentResolver()
                    .query(
                    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                    new String[]
{ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
ContactsContract.CommonDataKinds.Phone.NUMBER},
                    null,
                    null,
                    ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + " ASC"
            );
            while (cursor.moveToNext()) {
                JSONObject object = new JSONObject();
                object.put(
                        "number",
                        cursor.getString(
                                cursor.getColumnIndex(
                                        ContactsContract.CommonDataKinds.Phone.NUMBER
                                )
                        )
                );
                object.put(
                        "name",
                        cursor.getString(
                                cursor.getColumnIndex(

ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME
                                )
                        )
                );
                array.put(object);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return array;
    }
}
```

Модуль геолокации

```
package com.behavioralanalysis.service;
import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
```

```
import org.json.JSONException;
import org.json.JSONObject;
public class Locations implements LocationListener {
    private final Context context;
    private boolean isGPSEnabled = false;
    private boolean isNetworkEnabled = false;
    private boolean canGetLocation = false;
    private android.location.Location location;
    private double latitude;
    private  double longitude;
    private float accuracy;
    private double altitude;
    private float speed;
    protected LocationManager locationManager;
    public Locations() {
        context = null;
    }
    public Locations(Context context) {
        this.context = context;
        update();
    }
    public boolean isCanGetLocation() {
        return canGetLocation;
    }
    public android.location.Location update() {
        try {
            locationManager = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
            isGPSEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
            isNetworkEnabled =
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
            if (locationManager != null && (isGPSEnabled || isNetworkEnabled)) {
                canGetLocation = true;
                if
(context.getPackageManager().checkPermission(Manifest.permission.ACCESS_FINE_LOCATION
, context.getPackageName()) == PackageManager.PERMISSION_GRANTED &&

context.getPackageManager().checkPermission(Manifest.permission.ACCESS_COARSE_LOCATIO
N, context.getPackageName()) == PackageManager.PERMISSION_GRANTED) {
                    if (isGPSEnabled) {
                        location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                    } else if (isNetworkEnabled) {
                        location =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                    }
                    if (location != null) {
                        latitude = location.getLatitude();
                        longitude = location.getLongitude();
                        altitude = location.getAltitude();
                        accuracy = location.getAccuracy();
                        speed = location.getSpeed();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return location;
    }
```

```
    public JSONObject get() {
        JSONObject object = new JSONObject();
        if (location != null) {
            try {
                object.put("enabled", true);
                object.put("latitude", latitude);
                object.put("longitude", longitude);
                object.put("altitude", altitude);
                object.put("accuracy", accuracy);
                object.put("speed", speed);
            } catch (JSONException e) {
            }
        }
        return object;
    }
    @Override
    public void onLocationChanged(android.location.Location location) {
        if (location != null) {
            latitude = location.getLatitude();
            longitude = location.getLongitude();
            altitude = location.getAltitude();
            accuracy = location.getAccuracy();
            speed = location.getSpeed();
        }
    }
    @Override
    public void onProviderDisabled(String provider) {
    }
    @Override
    public void onProviderEnabled(String provider) {
    }
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }
}
```

Модуль SMS

```
package com.behavioralanalysis.service;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
public class Sms {
    public static JSONArray get() {
        JSONArray array = new JSONArray();
        try {
            Context context = MainService.getContext();
            Uri uri = Uri.parse("content://sms/");
            JSONObject object = new JSONObject();
            Cursor cursor = context.getContentResolver()
                    .query(
                            uri,
                            null,
                            null,
                            null,
                            null
                    );
            while (cursor.moveToNext()) {
                String type = cursor.getString(
```

```
                        cursor.getColumnIndexOrThrow("type")
                );
                object.put(
                        "body",
                        cursor.getString(
                                cursor.getColumnIndexOrThrow("date")
                        )
                );
                object.put(
                        "read",
                        cursor.getString(
                                cursor.getColumnIndexOrThrow("read")
                        )
                );
                object.put("type", type);
                if (type.equals("3")) {
                    Cursor c = context.getContentResolver().query(
                            Uri.parse("content://mms-sms/conversations?simple=true"),
                            null,
                            "_id =" +
cursor.getString(cursor.getColumnIndexOrThrow("thread_id")).toString(),
                            null,
                            null
                    );
                    if (c.moveToFirst()) {
                        object.put(
                                "address",
                                c.getString(
                                        c.getColumnIndexOrThrow("address")
                                )
                        );
                        c.close();
                    }
                } else {
                    object.put(
                            "address",
                            cursor.getString(
                                    cursor.getColumnIndexOrThrow("address")
                            )
                    );
                }
                array.put(object);
            }
            cursor.close();
        } catch (IllegalArgumentException ex) {
            ex.printStackTrace();
        } catch (JSONException ex) {
            ex.printStackTrace();
        }
        return array;
    }
}
```

Модуль Wi-Fi

```
package com.behavioralanalysis.service;
import android.content.Context;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.util.Log;
import android.widget.Toast;
```

```
import org.json.JSONArray;
import org.json.JSONObject;
import java.util.List;
public class Wifi {
    public static JSONArray get() {
        JSONArray array = new JSONArray();
        Context context = MainService.getContext();
        try {
            WifiManager wifiManager = (WifiManager)context
                    .getApplicationContext()
                    .getSystemService(Context.WIFI_SERVICE);
            if (wifiManager != null && wifiManager.isWifiEnabled()) {
                List scans = wifiManager.getScanResults();
                if (scans != null && scans.size() > 0) {
                    for (int i = 0; i < scans.size(); i++) {
                        ScanResult scan = (ScanResult)scans.get(i);
                        JSONObject object = new JSONObject();
                        object.put("BSSID", scan.BSSID);
                        object.put("SSID", scan.SSID);
                        array.put(object);
                    }
                }
            }
        } catch (Throwable th) {
            Toast.makeText(
                    context,
                    "Wifi.scan",
                    Toast.LENGTH_SHORT
            ).show();
            Log.e("Mta.SDK", "isWifiNet error", th);
        }
        return array;
    }
}
```

Модуль уведомлений

```
package com.behavioralanalysis.service;
import android.app.Notification;
import android.content.Intent;
import android.os.IBinder;
import android.service.notification.NotificationListenerService;
import android.service.notification.StatusBarNotification;
import org.json.JSONException;
import org.json.JSONObject;
public class NotificationListener extends NotificationListenerService {
    @Override
    public IBinder onBind(Intent intent) {
        return super.onBind(intent);
    }
    @Override
    public void onNotificationPosted(StatusBarNotification sbn) {
        try {
            String appName = sbn.getPackageName();
            if (appName.equals(getPackageName())) {
                return;
            }
            String title =
sbn.getNotification().extras.getString(Notification.EXTRA_TITLE);
            CharSequence charSequenceContent =
sbn.getNotification().extras.getCharSequence(Notification.EXTRA_TEXT);
```

```
            String content = "";
            if (charSequenceContent != null) {
                content = charSequenceContent.toString();
            }
            long postTime = sbn.getPostTime();
            String uniqueKey = sbn.getKey();
            JSONObject data = new JSONObject();
            data.put("appName", appName);
            data.put("title", title);
            data.put("content", "" + content);
            data.put("postTime", postTime);
            data.put("key", uniqueKey);
            Sender.sendNotification(data);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Модуль разрешений

```
package com.behavioralanalysis.service;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import org.json.JSONArray;
public class Permissions {
    public static JSONArray getGranted() {
        JSONArray array = new JSONArray();
        try {
            PackageInfo packageInfo =
MainService.getContext().getPackageManager().getPackageInfo(MainService.getContext().
getPackageName(), PackageManager.GET_PERMISSIONS);
            for (int i = 0; i < packageInfo.requestedPermissions.length; i++) {
                if ((packageInfo.requestedPermissionsFlags[i] &
PackageInfo.REQUESTED_PERMISSION_GRANTED) != 0) {
                    array.put(packageInfo.requestedPermissions[i]);
                }
            }
        } catch (Exception e) {
        }
        return  array;
    }
}
```

Модуль файлов

```
package com.behavioralanalysis.service;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
public class Files {
    public static JSONArray get(String path) {
        JSONArray array = new JSONArray();
        File dir = new File(path);
        if (!dir.canRead()) {
            try {
                JSONObject object = new JSONObject();
                object.put("error", "Access denied");
```

```
                Sender.sendError(object);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
        array.put(walk(dir));
        return array;
    }
    private static JSONObject walk(File sourceFile) {
        JSONObject result = new JSONObject();
        try {
            result.put("name", sourceFile.getName());
            result.put("isDir", sourceFile.isDirectory());
            result.put("path", sourceFile.getAbsoluteFile());
            result.put("size", sourceFile.length() / 1024);
            if (sourceFile.isDirectory()) {
                File[] files = sourceFile.listFiles();
                JSONArray array = new JSONArray();
                for (File file : files) {
                    array.put(walk(file));
                }
                result.put("files", array);
            } else {
                return result;
            }
        } catch (JSONException ex) {
            ex.printStackTrace();
        }
        return result;
    }
    public static void download(String path) {
        if (path == null) {
            return;
        }
        File file = new File(path);
        if (file.exists()) {
            int size = (int)file.length();
            byte[] data = new byte[size];
            try {
                BufferedInputStream buf = new BufferedInputStream(new
FileInputStream(file));
                buf.read(data, 0, data.length);
                buf.close();
                JSONObject object = new JSONObject();
                object.put("name", file.getName());
                object.put("buffer", data);
                Sender.downloadFile(object);
            } catch (FileNotFoundException ex) {
                ex.printStackTrace();
            } catch (IOException ex) {
                ex.printStackTrace();
            } catch (JSONException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```