1. Тема выпускной квалификационной работы специалиста

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

1. Цель и задачи выпускной квалификационной работы специалиста

Цель работы:

Задачи работы:

Вы

Выв

Выв

1. Этапы выпускной квалификационной работы специалиста
2. Перечень разрабатываемых документов и графических материалов

//

//

1. Руководитель выпускной квалификационной работы

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc61135260)

[ВВЕДЕНИЕ 4](#_Toc61135261)

[Перечень сокращение и условных обозначений 6](#_Toc61135262)

[Термины и определения 7](#_Toc61135263)

[1 Исследование предметной области 9](#_Toc61135264)

[1.1 Актуальность проблемы 9](#_Toc61135265)

[1.2 Анализ современных мобильных ОС 10](#_Toc61135266)

[1.2.1 Android 10](#_Toc61135267)

[1.2.2 iOS 11](#_Toc61135268)

[1.2.3 Windows Phone 12](#_Toc61135269)

[1.3 Выбор операционной системы для рассмотрения состояния защищенности мобильного устройства 13](#_Toc61135270)

[1.4 Выделение каналов утечки информации 13](#_Toc61135271)

[1.5 Принципы обнаружения вредоносных приложений для ОС Android 14](#_Toc61135272)

[1.5.1 Обнаружение вредоносных приложений для Android 14](#_Toc61135273)

[1.5.2 Методы, затрудняюшие криминалистический анализ 15](#_Toc61135274)

[1.5.2.1 Обфускация 15](#_Toc61135275)

[1.5.2.2 Шифрование символьных строк 16](#_Toc61135276)

[1.5.2.3 Проверка окружения 16](#_Toc61135277)

[1.6 Анализ следов вредоносного ПО 16](#_Toc61135278)

[1.6.1 Статический анализ вредоносного ПО 16](#_Toc61135279)

[1.6.1.1 Статический анализ. ApkTool 17](#_Toc61135280)

[1.6.1.2 Статический анализ. Dex2Jar 17](#_Toc61135281)

[1.6.2 Динамический анализ вредоносного ПО 17](#_Toc61135282)

[1.7 Этапы анализа следов предположительно вредоносного ПО в ОС Android 18](#_Toc61135283)

[1.8 Обзор средств проведения динамического анализа приложений для Android 18](#_Toc61135284)

[1.9 Выводы первого раздела 20](#_Toc61135285)

[2 Авав 22](#_Toc61135286)

[3 Экономическая оценка проекта 23](#_Toc61135287)

[3.1 Определение списка работ и состава исполнителей 23](#_Toc61135288)

[3.2 Расчет продолжительности и трудоемкости работ 24](#_Toc61135289)

[3.3 Планирование разработки программного обеспечения с построением диаграммы Ганта 28](#_Toc61135290)

[3.4 Расчет экономической эффективности 30](#_Toc61135291)

[3.4.1 Расчет затрат на основные материалы 30](#_Toc61135292)

[3.4.2 Заработная плата исполнителей 31](#_Toc61135293)

[3.4.3 Расчет накладных расходов 32](#_Toc61135294)

[3.4.4 Расчёт стоимости машинного времени 33](#_Toc61135295)

[3.4.5 Общая смета затрат на разработку проекта 33](#_Toc61135296)

[3.5 Расчет окупаемости программного обеспечения 34](#_Toc61135297)

[3.6 Выводы третьего раздела 36](#_Toc61135298)

[Выводы 37](#_Toc61135299)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc61135300)

[ПРИЛОЖЕНИЕ А 40](#_Toc61135301)

ВВЕДЕНИЕ

Большинство компаний в качестве корпоративных средств связи используют устройства на операционной системе андроид. К таким устройствам, в основном, относятся мобильные телефоны, которые используются внутри и вне предприятия для мобильной сотовой связи, обмена текстом и файлами по средствам смс, а также для выхода в интернет. Ввиду этого, важным аспектом эксплуатирования подобных устройств является мониторинг и обеспечение безопасности данных устройств, с целью предотвращения утечек информации.

В рамках данной работы будет проведен анализ существующих средств мониторинга безопасности мобильных устройств на операционной системе андроид, а также предложено собственное решение данной проблематики.

Актуальность данной работы заключается в том, что большинство используемых в компаниях средств связи работают на операционной системе андроид, так как такие устройства экономически более выгодны, при этом не оборудованы средствами мониторинга утечки данных.

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. предложить 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.

1. В первой главе …

Во второй главе …

В третьей главе …

Новизна работы:

1. выв

Практическая значимость

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

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

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

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

Десктоп – компьютер, предназначенный для постоянного размещения на столе.

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

Форензика – компьютерная криминалистика. Прикладная наука о раскрытии преступлений, связанных с компьютерной информацией, об исследовании цифровых доказательств, методах поиска, получения и закрепления таких доказательств.

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.

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.

Декомпиляция – процесс воссоздания исходного кода декомпилятором.

Дизассемблировоние – процесс восстановления ассемблерного кода.

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

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

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

HTTP – протокол прикладного уровня передачи данных.

SSL – криптографический протокол, который подразумевает более безопасную связь.

JSON – текстовый формат обмена данными, основанный на JavaScript,

SSID – символьное название беспроводной точки доступа Wi-Fi.

BSSID – MAC-адрес точки доступа Wi-Fi.

MAC – физическое имя устройства

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.

Логирование – запись данных о работе программы

Стек технологий – это набор инструментов, применяющийся при работе в проектах и включающий языки программирования, фреймворки, системы управления базами данных, компиляторы и т. д.

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

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

VPN – виртуальная частная сеть.

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.

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

1. Исследование предметной области
   1. **Актуал**ьно**сть проблемы**

Наиболее важна целостность приложений, содержащих личные данные или, как в случае коммерческого использования, коммерческую тайну.

Различают два типа анализа вредоносного ПО: статистический и динамический (поведенческий) [1]. 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, то есть приложений, 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. [2]

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. [3]

* 1. Анализ современных мобильных ОС

Мобильные устройства начали развиваться заметно позже стацио-нарных 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 и десктопных ОС – 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.

На сегодняшний день к списку наиболее популярных ОС для мо-бильных устройств можно отнести:

* Android;
* iOS;
* 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. Android

Android – операционная система, разрабатываемая компанией Android Inc., которая была выкуплена компанией Google в июле 2005 года [4]. 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 между народную статистику, предоставляемую сайтом StatCounter.com (рисунок 1), Android является наиболее популярной операционной системой в мире, составляя примерно 40% из всех используемых ОС, включая десктопные системы такие, как Windows [5]. Ввиду этого фактора можно сказать, что на текущий момент это самая популярная операционная система.

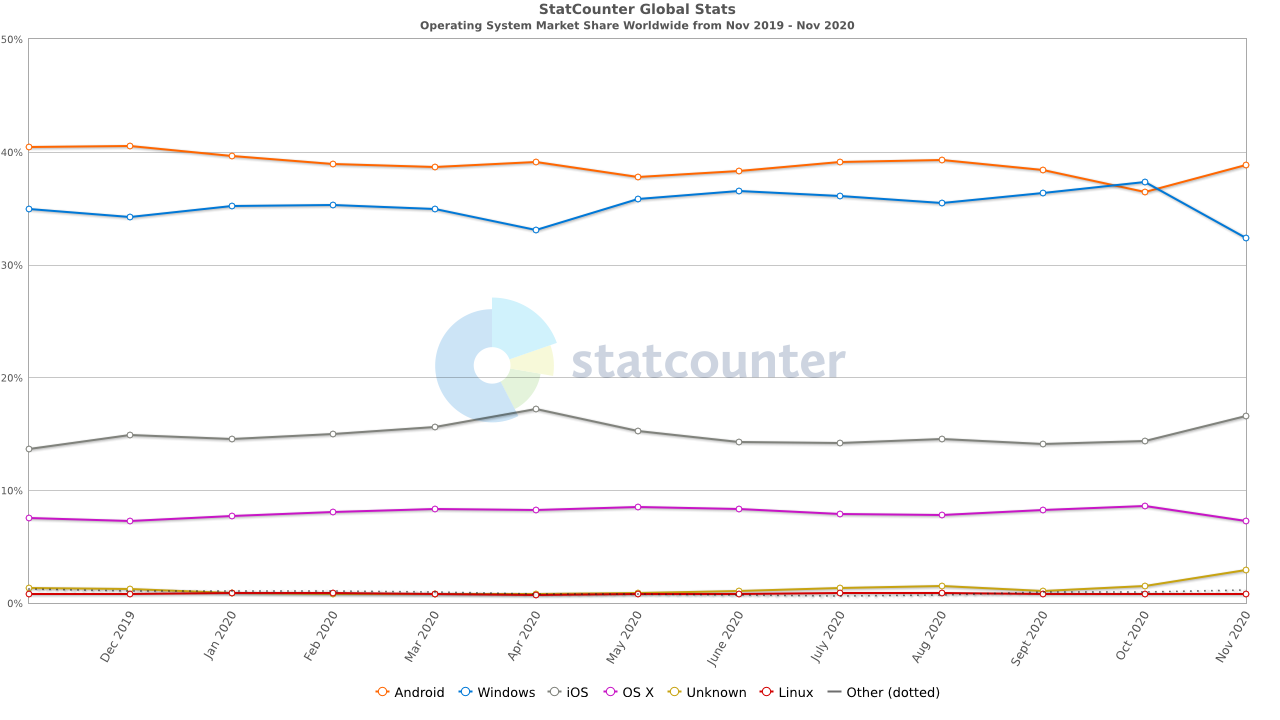


Рисунок 1 – Статистика популярности мобильных операционных систем

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. 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. [6]

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. 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. [7]

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.

* 1. Выбор операционной системы для рассмотрения состояния защищенности мобильного устройства

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 рассмотрения наиболее популярных мобильных ОС можно сделать вывод, что самой подходящей системой для поиска дополнительных средств обеспечения информационной безопасности является Android.

* 1. Выделение каналов утечки информации

Наиболее популярным каналом утечки корпоративной информации является инсайдер, то есть сотрудник организации, по средствам раскрытия закрытой информации внешнему нарушителю, с целью передачи конечной информации или предоставления ему несанкционированного доступа в систему из вне.

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

* 1. Принципы обнаружения вредоносных приложений для ОС Android

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. [8]

С точки зрения информационной безопасности, наиболее важные части APK-файла – сигнатура, байт-код и ресурсы.

* + 1. 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.

* + 1. Методы, затрудняющие криминалистический анализ

Существует четыре наиболее распространенных метода противодействия криминалистическому исследованию:

* Обфускация;
* Шифрование символьных строк;
* Противодействие декомпиляции;
* Проверка окружения.
  + - 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.

* + - 1. 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.

* + - 1. 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.

* 1. Анализ следов вредоносного ПО

Различают два типа анализа вредоносного ПО:

* Статический;
* Динамический (поведенческий).
  + 1. Статический анализ вредоносного ПО

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.

* + - 1. Статический анализ. ApkTool

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).
  + - 1. Статический анализ. Dex2Jar

Данное средство 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.

* + 1. Динамический анализ вредоносного ПО

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

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 осуществляе-мых вызовах.

В результате работы программы будет получен файл в формате JSON, содержащий вышеперечисленные сведения. [8]

* 1. Этапы анализа следов предположительно вредоносного ПО в ОС Android

Подводя итог по описанным методам анализа приложения для An-droid, 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.
   1. 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 дей-ствий 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 [9] 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 [10]. TaintDroid реализован на виртуальной машине Dalvik и мониторах заявки на утечку конфиденциальной информации. Однако, ScrubDroid [11] представил ряд атак для обхода динамического анализа заражения. VetDroid [12] – 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 и 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 изображена схема работы Crowdroid [13], инструмента для анализа поведения приложений. Это лёгкое клиентское приложение, которое мониторит системные вызовы, инициированные наблюдаемым приложением, предварительно обрабатывает их и отправляет в облако. А там уже 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, в первой версии 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. Чтобы этого избежать, будет введены механизмы шифрования, чтобы обеспечить целостность данных и подлинность отправителя.

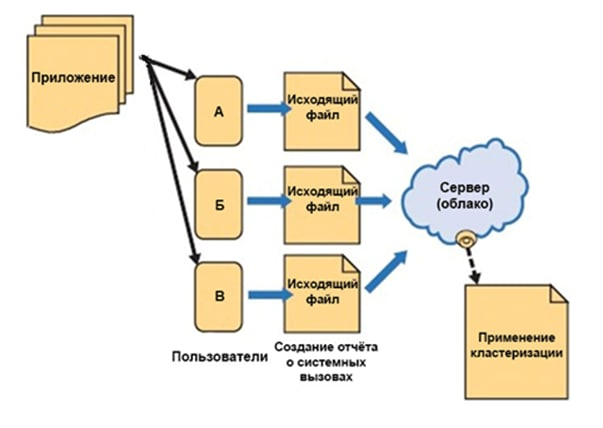


Рисунок 2 – Схема работы Crowdroid

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. Эти проблемы требуют некоторой ручной проверки или дальнейшего автоматического анализа. Во-вторых, можно намеренно отправить правильные данные в систему, оставив набор данных поврежденным.

* 1. Выводы первого раздела

На основе исследования рассмотренных примеров было выявлено, что наиболее популярными системами динамического анализа функционирования мобильных устройств являются средства изолированного запуска предполагаемо вредоносного ПО. Однако развиваются системы динамического анализа в режиме реального времени с использованием облачных сервисов, нейросетей и машинного обучения. Все системы имеют свои как преимущества, так и недостатки, но ни одна из представленных в свободном доступе не обеспечивает потребностей современной информационной безопасности, так как необходим непрерывный анализ и мониторинг действий в ОС в режиме реального времени.

1. Сервис поведенческого анализа функционирования мобильных устройств на операционных системах андроид
   1. Разработка архитектуры сервиса

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

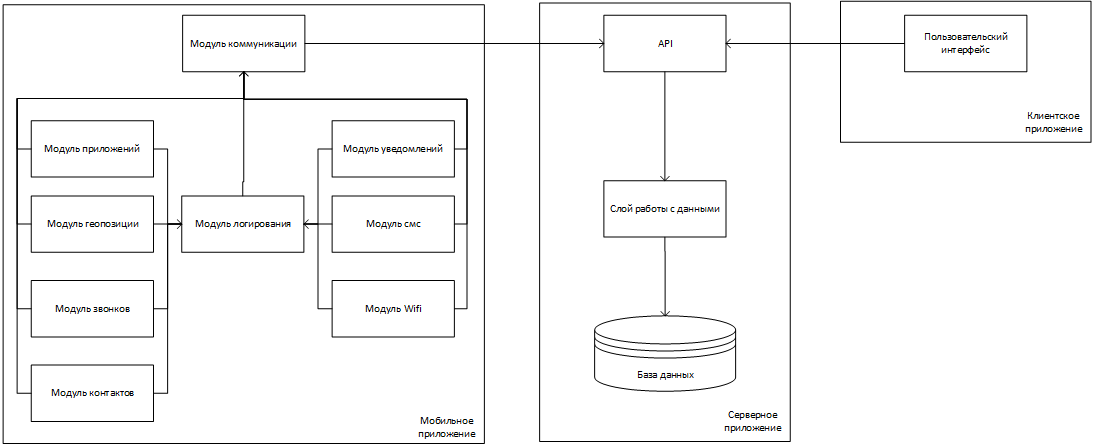


Рисунок 3 – Архитектура сервиса

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. 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.

* 1. 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 [14], в рамках которого, с помощью языка программирования C# было реализовано серверное приложение данного сервиса;
* React – JavaScript-библиотека для разработки пользовательских интерфейсов [15], с помощью которой на языке программирования TypeScript было написано клиентское приложение;
* IIS – набор сервисов для служб интернета [16], 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 [17];
* Microsoft Visual Studio 2019 – интегрированная среда разработки программного обеспечения и ряд других инструментальных средств [18];
* Microsoft Visual Studio Code – редактор исходного кода, разработанный Microsoft [19].

Разработка системы, и ее тестирование производилось на устройствах, чьи характеристики представлены в таблице 2.1.

Таблица 2.1 – 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 |

* 1. 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.
  + 1. 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.

* + 1. 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 2.3.1, последовательно запускается каждый из модулей.

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.

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.

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.

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.

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.

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.

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.

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

Каждый из модулей 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.

* + 1. 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.

* + 1. 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.

* + 1. 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.

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

* 1. 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

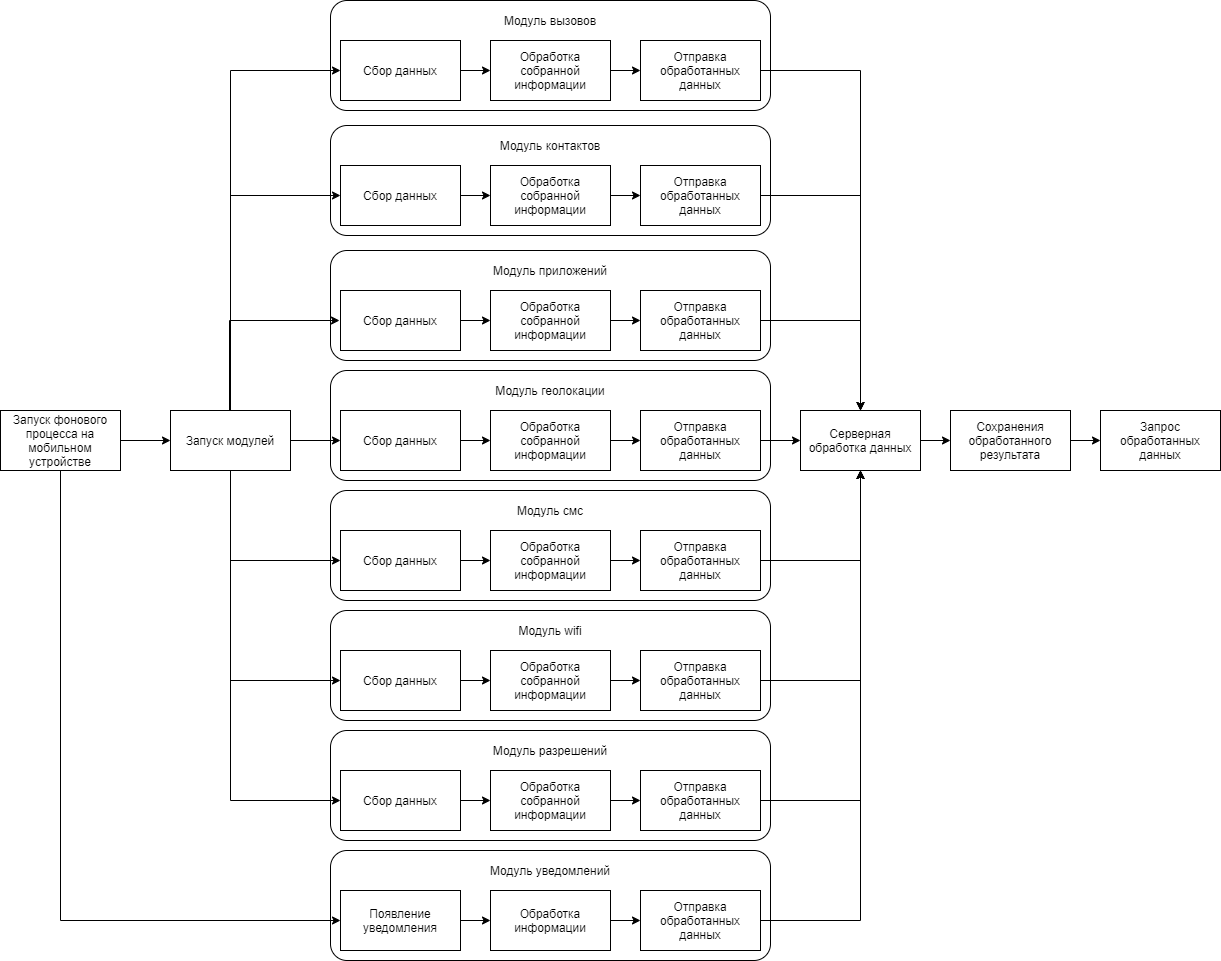


Рисунок 4 – Общий алгоритм работы системы

* 1. Выводы второго 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.

1. 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.

* 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 3.1, приведенной ниже, содержится состав исполнителей и полный перечень работ.

Таблица 3.1 — 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 |
| 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 |

Продолжение таблицы 3.1

* 1. Расчет 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 (3.1):

где ti – трудоемкость работ, человек-дней;

tmin – минимально возможная трудоемкость выполнения отдельного вида работ в человеко-днях;

tmax – максимально возможная трудоемкость выполнения отдельного вида работ в человеко-днях.

Продолжительность каждого вида работ в календарных днях (Ti) определяется по формуле (3.2):

где ti – трудоемкость работ, человек-дней;

Чi – численность исполнителей, человек;

Kвых – коэффициент, учитывающий выходные и праздничные дни, определяющийся по формуле (3.3):

,

, где Kкал – число календарных дней;

Kраб – рабочие дни;

Для расчёта принимается среднее значение Kвых = 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 3.2.

Таблица 3.2 – Расчёт трудоёмкости и продолжительности работ по созданию ПО.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № работы | Наименование | Трудоемкость, чел. часы | | | Количество исполнителей, чел. | Продолжительность, календарные дни |
| стадий и работ | *t*min | *t*max | ti | Чi | Тi |
| Teхнический 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 |
| 15 | Согласование и утверждение технического проекта | 1 | 2 | 1.4 | 1 | 2.1 |
| Рабочий проект | | | | | | |
| 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.

* 1. 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 3.2 приведена на рисунке 3.1.

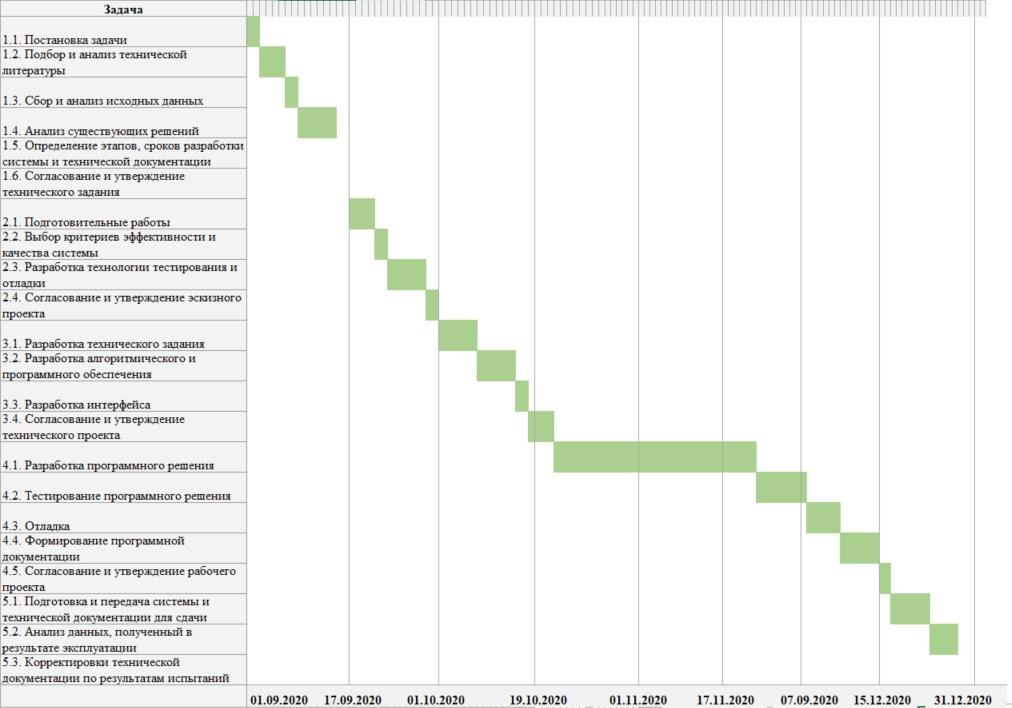


Рисунок 3.1 – Диаграмма Ганта

* 1. Расчет экономической эффективности

В выпускной квалификационной работе величину затрат на разработку (Cразр) программного продукта можно найти по формуле 3.4:

,

где Сразр– стоимость разработки программного средства;

Смат – затраты на используемые материалы;

Cосн – заработная плата исполнителей;

Cдоп – дополнительная заработная плата исполнителей (в среднем берется около 15% от Сосн);

Сотч – отчисления от заработной платы исполнителей (пенсионный фонд, фонд обязательного медицинского страхования, фонд социального страхования, принимается, примерно, как 30% от Сосн и Сдоп;

Cнакл – накладные расходы (уборку, ремонт, и др., берется в размере около 60% от Сосн и Сдоп);

Cм.вр – затраты машинного времени.

* + 1. 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 3.5

,

где Р – потребляемая мощность оборудования, кВт/ч;

Цэл – стоимость 1 кВт/ч, руб.;

Ти – время использования оборудования при проведении работ, ч.

,

Эти затраты определяются экспертным путем. Пример расчета расходов показан в таблице 3.3.

Таблица 3.3 – 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](https://www.komus.ru/katalog/tekhnika/kompyutery-i-periferiya/nositeli-informatsii/flesh-pamyat-usb/flesh-pamyat-transcend-jetflash-350-4-gb-usb-2-0-chernaya/p/272691/?from=block-123-1) | кВт/час | 5,47 | 52,35 | 286,4 |
| 3 | Домен | Штука/год | 300 | 1 | 300 |
| 4 | Другие канцтовары | – | – | – | 523,6 |
| 5 | Итого, Смат | | | | 1345 |

* + 1. 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 3.6:

,

где Сосн – заработная плата исполнителей (руб.);

Т – трудоёмкость разработки (человеко-дни);

Сср – средняя дневная тарифная ставка работника организации разработчика проекта(руб./человеко-дни), которая определяется по формуле 3.7:

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

Фмес – среднее количество рабочих дней в месяце (20 дней).

Дополнительная заработная плата на период разработки вычисляется по формуле 3.7:

Расчёт затрат на основную заработную плату разработчиков проекта приведен в таблице 3.4.

Таблица 3.4 – 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 3.8, затраты на внебюджетные фонды составляют:

* + 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 3.9 расчёта представлена ниже:

,

где Снакл – накладные расходы (руб.);

Сосн – основная заработная плата исполнителей (руб.);

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

Итак, в нашем случае:

* + 1. 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 (3.10):

*,*

где – стоимость одного часа машинного времени. Допустим, что в данной ситуации = 3 руб/час;

– машинное время, которое необходимо для создания проекта. Оно рассчитывается по формуле (3.11):

,

где – продолжительность рабочей смены ( 8 часов – длительность рабочего дня)

– это средний коэффициент использования машинного времени.

Большая часть проекта непосредственно связана с работой за компьютером (работа с документацией в Word и разработка), поэтому возьмем равной за 0,9.

Получим:

В результате расчетов можно вычислить итоговую себестоимость анализа проекта с учетом внедрения утилиты, которая рассчитывается по формуле (3.4). Рассчитаем себестоимость:

* + 1. Общая смета затрат на разработку проекта

Результаты расчета затрат на разработку программного обеспечения для информационного поиска и классификации текстов представлены в таблице 3.5.

Таблица 3.5 – Смета затрат на разработку проекта.

|  |  |  |  |
| --- | --- | --- | --- |
| **Статья затрат** | **Обозначение** | **Величина затрат (руб.)** | **% затрат к итогу** |
| **Затраты на основные материалы** | Смат | 1345.00 | 0.29% |
| **Основная заработная плата** | Сосн | 222956.00 | 47.40% |
| **Дополнительная заработная плата** | Сдоп | 33443.40 | 7.11% |
| **Отчисления от заработной платы** | Сотч | 76919.90 | 16.35% |
| **Накладные расходы** | Снакл | 133773.80 | 28.44% |
| **Машинное время** | См.вр | 1963.44 | 0.42% |
| **Итого** | Сразр | 470401.54 | 100,00% |

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

* 1. Расчет окупаемости программного обеспечения

Данная программа может быть реализована на рынке. При расчётном количестве реализованных программ – n, оптовая цена программы (Цопт) может быть рассчитана по формуле 3.12:

,

где Сразр – себестоимость разработки программы;

П – прибыль, определяется по формуле 3.13:

,

где Yр – средний уровень рентабельности, принимается Yр= 20%;

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

Отпускная оптовая цена реализации программы потребителям должна включать налог на добавленную стоимость (НДС) и рассчитывается по формуле 3.14:

,

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

НДС = 5644,8 руб.

Цопт = 28224,1 руб.

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

Затраты на внедрение системы () 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.

руб.

Итак, затраты на сопровождение системы (Ссопр) зависят только от затрат на оплату труда. Однако покупка пакета поддержки не является обязательной услугой.

руб.

Для расчета срока окупаемости систем используют формулу 3.15:

,

где T – срок окупаемости системы;

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

мес.

* 1. Выводы третьего 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.

Выводы

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Монаппа К. А. Анализ вредоносных программ / пер. с англ. Беликова Д. А. – М.: ДМК Пресс, 2019. – 452с.
2. [Электронный ресурс]: Positive Technologies. – URL: <https://www.ptsecurity.com/ru-ru/research/analytics/financial-application-vulnerabilities-2018/> (дата обращения 15.10.2020)
3. [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)
4. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Android> (дата обращения 21.10.2020)
5. [Электронный ресурс]: Statcounter GlobalStats. – URL: <https://gs.statcounter.com/os-market-share> (дата обращения 21.10.2020)
6. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/IOS> (дата обращения 21.10.2020)
7. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Windows_10_Mobile> (дата обращения 21.10.2020)
8. [Электронный ресурс]: Anti-Malware. – URL: <https://www.anti-malware.ru/analytics/Threats_Analysis/principles_detection_malicious_applications_Android> (дата обращения 22.10.2020)
9. 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
10. 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
11. 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
12. 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
13. 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
14. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/.NET_Core> (дата обращения 10.01.2021)
15. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/React> (дата обращения 10.01.2021)
16. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Internet_Information_Services> (дата обращения 10.01.2021)
17. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Android_Studio> (дата обращения 10.01.2021)
18. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2019> (дата обращения 10.01.2021)
19. [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Visual_Studio_Code> (дата обращения 10.01.2021)

ПРИЛОЖЕНИЕ А