# Аннотация

В данном дипломном проекте была рассмотрена проблема безопасности операционной системы Android и программного обеспечения, созданного для этой ОС. На основе проведенного исследования было выявлено узкое место ОС, представляющее для пользователя вполне реальную опасность. Выявлена и обоснована необходимость создания программных решений, направленных на разъяснение пользователю чем чревато использование того или иного ПО на Android устройстве. Была предложена архитектура системы, позволяющая повысить уровень безопасности ОС Android и способная дать оценку опасности приложению для указанной мобильной платформы. Так же были даны оценки эффективности предложенного решения.

Содержание

[Аннотация 1](#_Toc357936928)

[Введение 3](#_Toc357936929)

[Раздел 1. Общий обзор проблемы безопасности ОС Android. Постановка задачи на дипломное проектирование. 6](#_Toc357936930)

[1.1 Развитие индустрии по борьбе с вредоносным и мошенническим ПО 6](#_Toc357936931)

[1.2 Описание предметной области 7](#_Toc357936932)

[1.3 Постановка задачи на дипломное проектирование, требования к системе 8](#_Toc357936933)

[1.4 Входные и выходные данные 9](#_Toc357936934)

[1.5 Система безопасности операционной системы Android 9](#_Toc357936935)

[1.6 Способы преодоления безопасности защиты ОС Android 16](#_Toc357936936)

[1.7 Типы опасных объектов 18](#_Toc357936937)

[1.8 Структура Android приложений 22](#_Toc357936938)

[1.9 Анализ существующих систем определения уровня опасности. 27](#_Toc357936939)

[1.10 Выводы. 31](#_Toc357936940)

[Раздел 2. Проектирование системы. 33](#_Toc357936941)

[2.1 Анализ и выбор языка программирования для проектирования Системы 33](#_Toc357936942)

[2.2 Разработка Системы ранжирования уровней опасности Android приложений 35](#_Toc357936943)

[2.2.1 Программные средства разработки 35](#_Toc357936944)

[2.2.2 Аппаратная среда разработки 35](#_Toc357936945)

[2.2.3 Алгоритм функционирования Системы 35](#_Toc357936946)

[2.2.4 Выбор производителя и типа СУБД. Проектирование структуры базы данных. 38](#_Toc357936947)

[2.2.5 Механизм ранжирования приложений по метаданным и по исполняемому коду. 39](#_Toc357936948)

[2.2.6 Интерфейс пользователя. 41](#_Toc357936949)

[2.2.7 Схема работы модулей системы 43](#_Toc357936950)

[2.3 Выводы 44](#_Toc357936951)

[Раздел 3. Оценка эффективности принятых решений 45](#_Toc357936952)

[3.1 Выбор и обоснование оценок эффективности. 45](#_Toc357936953)

[3.2 Выбор и обоснование методики тестирования 45](#_Toc357936954)

[3.3 Тестирование разработанного программного обеспечения 47](#_Toc357936955)

[3.3.1 Требования к Системе 47](#_Toc357936956)

[3.3.2 Средства тестирования 50](#_Toc357936957)

[3.3.3 Результаты тестирования системы 50](#_Toc357936958)

[3.4 Оценка и результатов 51](#_Toc357936959)

[3.7 Надежность разрабатываемой системы. 52](#_Toc357936960)

[3.8 Выводы 57](#_Toc357936961)

[Заключение 58](#_Toc357936962)

[Список литературы 60](#_Toc357936963)

[Приложение 1. Руководство пользователя 62](#_Toc357936964)

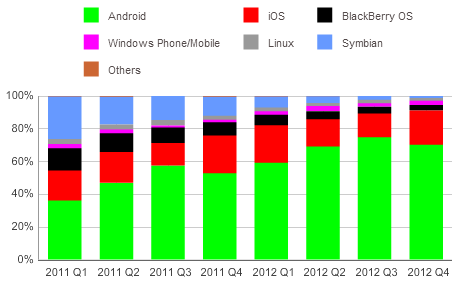
[Приложение 2. Распечатка текстов программных модулей 65](#_Toc357936965)

# Введение

Современное общество невозможно представить без портативных компьютеров, мобильных телефонов, планшетов и прочих устройств.

По данным IDC, квартал от квартала доля использования операционной системы Android только растёт и в конце 2012 года достигает 75%

На Рисунке 1 представлена картина динамики роста доли проданных устройств на базе платформы Android за последние два года.



**Рисунок 1. Доля Android на фоне других мобильных операционных систем**

Подавляющее большинство устройств на ОС Android составляют смартфоны. Являясь большей частью компактным персональным компьютером, смартфон дополнен функциональностью мобильного телефона.

Смартфоны отличаются от обычных мобильных телефонов наличием операционной системы Android. Данная платформа позволяет любому разработчику создать приложение и начать его продавать в специальном магазине приложений Google Play. Так как «из коробки» устройства на базе Android содержат объективный минимум предустановленных приложений, то использование магазина и сторонних разработчиков приложений вместе с их разработками позволяет по достоинству раскрыть возможности операционной системы Android.

Следует заметить, что в борьбе за покупателя производители смартфонов всё больше и больше снижают себестоимость устройств. Благодаря этому смартфон можно приобрести по бросовой цене и подарить его сыну или бабушке.

Цифры статистики подтверждают тот факт, что смартфоны на ОС Android есть в каждой семье и у детей и у родителей.

Производители электроники внедряют мобильные операционные системы в бытовые устройства — холодильники, стиральные машины. Разработку «умной техники», способной передавать данные и команды через интернет, ведут уже более 30 компаний. Пока они отдают предпочтение операционной системе Android, которая доступнее платформ Microsoft и Apple.

Производители отдают предпочтение Android из-за отсутствия платы за использование и простоту внедрения платформы. Первые смартфоны на базе Android поступили в продажу в 2008 году, и в третьем квартале 2012 года их доля на рынке составила 72%, подсчитали в Gartner.

Разумеется, чтобы использовать возможности операционной системы максимально необходимо

1. Вывести устройства в сеть Internet
2. Воспользоваться возможностями магазина приложений Google Play
3. Воспользоваться возможностями электронных платежей и Internet-банкинга

Все три фактора в той или иной степени связаны с привязкой устройства к конфиденциальным данным пользователя, в том числе к его банковскому счету.

Поэтому, проблема защиты для Android устройств является очень серьезной и актуальной.

На данный момент существуют множество решений обеспечивающих защиту мобильных устройств. Все эти средства базируются на детектировании, в том числе вредоносного и потенциально нежелательного ПО. При этом детектирование строится исходя из конкретного функционала. Как только этот функционал преобразуется в похожий – детектирование прекращается. Исходя из этого появилась необходимость в системе, способной определить уровень опасности независимо от детектирования тем или иным Антивирусным программный обеспечением, принимая во внимание только структуру приложения, его исполняемых файлов.

# Раздел 1. Общий обзор проблемы безопасности ОС Android. Постановка задачи на дипломное проектирование.

## 1.1 Развитие индустрии по борьбе с вредоносным и мошенническим ПО

«Мы здесь, чтобы спасти мир. Мы считаем, что каждый должен иметь возможность получать от технологий максимум, не опасаясь при этом вторжения в личное цифровое пространство. Благодаря команде наших специалистов вы можете обитать в цифровом мире, не тревожась за личную информацию и финансовые средства. Мы разрабатываем, производим и продаем решения, обеспечивающие защиту наших клиентов от информационных угроз и позволяющие компаниям управлять IT-рисками»

Евгений Касперский, генеральный директор и один

из учредителей «Лаборатории Касперского».

Число вредоносного и потенциально опасного ПО постоянно растёт и владельцы устройств на базе ОС Android уже не могут использовать свой смартфон без защитного ПО. За последние пару лет число Антивирусных компаний, которые выпустили свои решения для мобильной платформы Android увеличилось в 5 раз. Теперь в число этих компаний помимо таких больших корпорации как Symantec, McAfee, TrendMicro входят и мало известные производители, такие как Aegis, AhnLab.

Не осталась и в стороне компания «Лаборатория Касперского»

Главное достояние компании — это богатый опыт, накопленный за годы непрерывной борьбы с самыми серьезными IT-угрозами и включающий антивирусные исследования, противодействие потенциально опасным приложениям, фильтрацию трафика и т.д. Благодаря этому компания всегда на шаг впереди конкурентов и обеспечивает пользователей самой надежной защитой от новых видов атак.

## 1.2 Описание предметной области

В данной работе идёт речь о приложениях, то есть программном обеспечении, созданном для запуска исключительно в среде операционной системы Android. Так как сама система обладает рядом особенностей и ограничений разработка приложений для неё весьма специфична – например, нельзя воспользоваться той или иной функции ОС не запросив у пользователя разрешения на такую активность.

Операционная система Android проектировалась с широким набором защитных функций, но из-за особенности распространения приложений, при которой нет никакого контроля за разработчиками и самими приложениями, появились случаи злоупотребления возможностями ОС.

Многие ведущие Антивирусные компании, предполагая что печальная ситуация с операционной системой семейства Microsoft Windows может перекочевать и на мобильные операционные системы, выпустили Антивирусные решения и под мобильные ОС в том числе и для Android OS.

В свою очередь вирусописатели, оценив колоссальный рост числа устройств с мобильной ОС Android начали создавать вредоносные приложения. Примером такого приложений может служить семейство Trojan-SMS. Приложения этого класса без ведома пользователя отсылают короткие SMS сообщения на номера со специальной тарифной сеткой. Одно такое короткое сообщение может лишить пользователя от 50 до 300 рублей.

В мою зону ответственности как Старшего вирусного аналитика «Лаборатории» Касперского входит анализ и детектирование приложений такого рода.

В данной работе будет вестись проектирование Системы, которая сама умеет определять насколько опасно то или иное приложение на основе знаний полученных на моём месте работы.

## 1.3 Постановка задачи на дипломное проектирование, требования к системе

Целью настоящей дипломной работы улучшить общую защиту ОС Android при условии установки приложения из стороннего (отличного от Google Play) источника при этом предоставить пользователю адекватную картину уровня опасности устанавливаемого приложения.

**Требования к системе:**

* определение уровня опасности по метаданным приложения
* определение уровня опасности по исполняемому файлу, формата DEX.
* сбор и хранение результатов ранжирования уровней опасности
* графическое отображение результатов на экран пользователя
* сохранение результатов в файл для дальнейшей постобработки

Для достижения поставленной цели необходимо решение следующих задач

* понять структуру безопасности ОС Android
* понять подходы к преодолению системы безопасности,
* изучить типы опасных объектов,
* проанализировать структуру установочного пакета приложения,
* выявить те области пакета, которые содержат информацию о приложении. На основе этой информации будет вестись определение уровня опасности.
* выбрать на каком языке программирования необходимо проектировать Систему,
* спроектировать систему, а именно выбор программный и аппаратных средств реализации, сформировать алгоритм работы системы.

## 1.4 Входные и выходные данные

*Входные данные:* На вход системе подаётся установочный пакет приложения для ОС Android. Данный пакет имеет специфическую структуру: формат пакета – архив типа ZIP, состоит как минимум из двух файлов: classes.dex и AndroidManifest.xml.

Если при работе системы в неё будет загружен установочный пакет иного формата, то система должна сообщить об ошибке.

Формат файла classes.dex бинарный, для его обработки необходимо разработать специальный статический анализатор.

*Выходные данные:*  Результатом работы системы должен файл отчета содержащий следующие поля:

* Название анализируемого файла
* Имя установочного пакета
* Хеш-сумма MD5, посчитанная от установочного пакета
* Числовое значение уровня опасности приложения с обозначенной рядом шкалой и словесным описанием уровня.
* Список словестных описаний полученного уровня.

Данные так же дублируются на дисплей пользователя в виде отдельного окна интерфейса Системы.

## 1.5 Система безопасности операционной системы Android

Android является современной мобильной платформой, которая была разработана, чтобы быть по-настоящему открытой. Android-приложения используют современное оборудование и программное обеспечение. Для защиты платформа должна предлагать прикладную среду, которая обеспечивает безопасность пользователей, данных, приложений, устройства и сети.

Android была разработана так чтобы снизить вероятность нападений и значительно ограничить влияние атаки в случае, если она был успешной.

Существуют два основных источника для приложений:

**Предустановленные приложения**: Android включает в себя набор предустановленных приложений, включая телефон, электронную почту, календарь, веб-браузер, и контакты. Эти функции в качестве пользовательских приложений обеспечивают основные возможности устройства, которые могут быть доступны для других приложений. Предустановленные приложения могут быть частью платформы с открытым кодом или они могут быть разработаны сторонними разработчиками для конкретного устройства.

**Приложения, установленные пользователем**: Android предоставляет открытую среду разработки и поддержки любого стороннего приложения. Google Play предлагает пользователям сотни тысяч приложений.

Google предоставляет набор облачных сервисов, которые доступны на любом совместимом устройстве Android. Основными услугами являются:

**Google Play:** Google Play представляет собой набор сервисов, которые позволяют пользователям находить, устанавливать и покупать приложения из своего устройства Android. Google Play делает путь создания приложений легким для разработчиков, чтобы достичь Android пользователей и потенциальных клиентов. Google Play также предоставляет средства анализа, проверки сертификатов о выдаче лицензии, приложение для сканирования безопасности и другие службы безопасности.

Архитектура системы безопасности Android:

Android стремится быть самой безопасной и пригодной для использования операционной системой для мобильных платформ. Это достигается путём динамического изменения целей традиционных элементов управления операционной в сторону системы безопасности, такими целями являются:

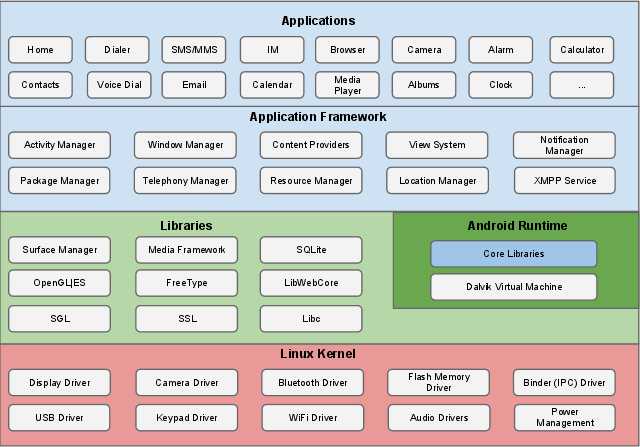
* Защитить данные пользователя
* Защита системных ресурсов (в том числе сети)
* Обеспечить изоляцию приложений

Для достижения этих целей, Android предоставляет следующие ключевые функции безопасности:

* Надежная система безопасности на уровне операционной системы через Linux ядро
* Обязательная изоляция всех приложений в специальной программной среде (песочнице)
* Безопасные связи между процессами
* Приложение должно быть подписанным сертификатом разработчика
* Разрешение на использование тех и или иных функций должно подтверждаться пользователем.

На Рисунке 2 представлена общая схема компонентов защиты с отображениями по различным уровням стека программного обеспечения Android.

Каждый компонент предполагает, что нижние компоненты были надежно защищены. За исключением небольшого количества кода Android OS мало что запущено с правами администратора. Все, что выше кода ядра Linux, ограничено изолированной программной средой.



**Рисунок 2. Программный стек ОС Android**

**Система безопасности приложений на основе запрашиваемых разрешений.**

Все приложения на Android работают в изолированной программной среде (песочнице). По умолчанию приложение Android может получить доступ только к ограниченному кругу системных ресурсов. Система сама управляет доступом к ресурсам, которые могут быть использоваться неправильно или злонамеренно, а так же могли бы негативно повлиять на пользователей, сети, или данные на устройстве.

Эти ограничения применяются в различных формах. Некоторые возможности ограничены преднамеренно путем исключения прямых интерфейсов к чувствительной функциональности устройства (например, не существуют прямых API функций для непосредственного использования SIM карты). В некоторых случаях разделение ролей обеспечивает меры безопасности на ряду с разделением хранимых данных приложения. В других случаях, чувствительные интерфейсы, предназначенные для использования доверенными приложениями, защищены через особый механизм безопасности известный как «разрешения».

В список защищенных разрешениями API функций входят:

* функции камеры
* Данные о местоположении (GPS)
* Функции Bluetooth
* Функции телефонии
* SMS / MMS функции
* Функции Сеть / передачи данных

Эти ресурсы напрямую доступны только через операционную систему. Чтобы использовать защищенные API, приложение должно определить необходимые возможности в своём манифесте. При подготовке к установке приложения, система выводит диалоговое окно для пользователя, в котором указывает разрешения и спрашивает согласия пользователя чтобы продолжить установку. Если пользователь даёт разрешение на установку то система считает, что пользователь предоставил все необходимые разрешения. Пользователь не может разрешить или запретить индивидуальные права доступа - пользователь должен разрешить или запретить все необходимые разрешения в виде блока.

После принятия разрешений и установкой приложения, принятые разрешения действуют пока приложение установлено в системе. Чтобы избежать путаницы пользователей, система не уведомит пользователя заново о предоставленных разрешениях, так же не будет уведомлений о приложениях, которые входят в ядро операционной системы или в комплекте по OEM. Разрешения будут удалены, если приложение будет удалено, так что последующая переустановка будет снова приведёт к отображение запроса разрешений.

В настройках устройства пользователи могут просмотреть разрешения для приложений, которые они ранее уже установили. Пользователи могут также отключить некоторые функции глобально, например отключить GPS, радио, или Wi-Fi.

В случае если приложение пытается использовать защищенную функцию, которая не была объявлена ​​в манифесте приложения, то такое поведение классифицируется как исключительное и выводится сообщение об ошибке. Проверки разрешения на доступ к защищённым API введены в действие на низком уровне, чтобы предотвратить обход. Пример доступа к защищенным API представлен на Рисунке 3:

|  |  |
| --- | --- |
| **Разрешения во время установки приложения** | **Разрешения уже установленного приложения** |
| Permissions at Application Install -- Google Maps | Permissions of an Installed Application -- gMail |

**Рисунок 3. Пример отображения разрешений на доступ к защищённым API**

Некоторые API функции объединены в особые группы:

**API Функции, чувствительные к цене**

Речь идет о функциях, использование которых может повлечь дополнительные траты со стороны пользователя и сети оператора. Пользователь должен будет предоставить явное разрешение для сторонних приложений, запрашивающих разрешение на использование чувствительных API. Эти API включают в себя:

* Телефония
* SMS / MMS
* Сеть / Internet данные
* Билинг внутри приложений
* NFC доступ

В Android 4.2 добавляется дополнительный контроль по использованию SMS. Android предоставит уведомление если приложение пытается отправить SMS на короткий номер, который используется для доступа к ПРЕМИУМ сервисам, которые могут привести к дополнительной оплате. Пользователь может выбрать, следует ли разрешить приложению отправлять сообщение или заблокировать его.

**Доступ к SIM карте**

Низкоуровневый доступ к SIM-карта не доступен для сторонних приложений. ОС обрабатывает все связи с SIM-картой, в том числе доступом к личной информации (контактам) в памяти на SIM-карте. Приложение также не может получить доступ к AT командам модема, так как они управляются исключительно их слоя радио интерфейса (RIL). RIL не предоставляет высокоуровневого доступа к этим командам.

**Персональная информация**

В Android все функции, работающие с личной информацией помещены в специальный набор защищенных API. При нормальных условиях эксплуатации, в Android устройствах накапливаются данные пользователя в сторонних приложениях, установленных пользователями. Приложения, которые решили поделиться этой информацией могут использовать механизм проверки прав доступа для защиты данных от сторонних приложений (проверка разрешений).

На Рисунке 4 представлена схема взаимодействия приложения с личной информацией пользователя, чувствительными устройствами ввода а так же метаданными устройства через разграничение разрешений.

Figure 3: Access to sensitive user data is only available through protected
APIs

**Рисунок 4. Взаимодействие приложения с пользовательскими данными.**

## 1.6 Способы преодоления безопасности защиты ОС Android

В большинстве случае для того чтобы совершить то или иное злонамеренное действие с использованием устройства на базе ОС Android необходимо проникнуть на само устройство(не считая атак Man-In-the-Middle), существует две основные тактики проникновения:

* социальная инженерия – использования влияния контента на психику;
* заражение чистого приложения, грязной составляющей.

Часто эти способы используются одновременно. При этом так же часто используются специальные меры по противодействию антивирусным программам.

**Социальная инженерия**

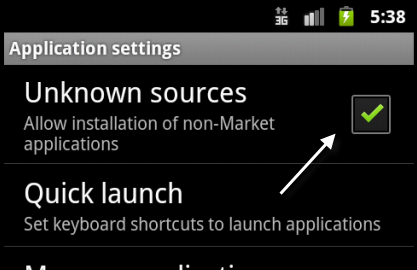
В данном случае речь идёт о психической атаке на пользователя – создать все условия (обмануть жертву) для того чтобы пользователь совершил необходимые действия.

Суть атаки заключается в создании специальных ресурсов - клонов (например, клонов легальных интернет магазинов) на которых распространяется программное обеспечение для Android. Из-за того что система позволяет ставить приложения из сторонних источников установить с ресурса-клона вредоносное ПО не составит труда. Для того чтобы заставить пользователя попасть на это ресурс существуют специальные рассылки. Наиболее распространённым случаем является получаемое пользователем короткое SMS сообщение вида «Привет. Я разместила свои фото тут [http://XXXX»](http://XXXX). Практика показывает что 90% получивших такое сообщение обязательно запустят файл на установку.

В данной технике вирусописатели постоянно пытаются предсказать поведение пользователя и выдать ему наиболее интересную ему информацию, внутри которой содержится опасный объект.

Но в силу особенности ОС Android пользователю, при установке приложения из стороннего источника (любой источник отличный от магазина приложений Google Play является сторонним), необходимо отключить часть системы безопасности ОС таким образом позволив проложить приложению процесс установки.

На Рисунке 5 представлено место в настройках ОС которое пользователю предложить активировать, в противном случае приложение из сторонних источников не установится.



**Рисунок 5. Подтверждение установки приложения из стороннего источника.**

**Внедрение вредоносного кода**

Данная техника инфицирования во многом сопряжена с применением социальной инженерии хотя бы потому, что необходимо подобрать наиболее популярное приложение – тем самым сделать число инфицированных пользователей максимальным.

Суть техники внедрения заключается в структуре исполняемого файла приложения для Android. При наличии специального ПО можно почти полностью декомпилировать приложение и затем его скомпилировать обратно, внедрив чужеродный код.

Затем приложение выкладывается в магазин под тем же именем, тем самым обходится проблемы недоверия пользователя. Затем пользователь ставит приложение, не принимая во внимания запрошенные разрешения. Таким образом работает два механизма одновременно.

## 1.7 Типы опасных объектов

Большинство приложений не содержат в себе вирусы в общепринятом понятии, но они способны к различному поведению, и могут создавать угрозу пользователю. Важно знать и то, что самостоятельно пользователю практически невозможно отличить опасное приложение от безопасного.

Опасные, приложения имеют возможность находить уязвимости в системе безопасности мобильных устройств. Недостатки кода приложения косвенно может способствовать тому, что хакерам удастся отследить телефонный номер пользователя, изменить закладки и добавить нежелательный контент в планшет или смартфон.

Как показало исследование, которое проводилось группой специалистов из Государственного Университета Северной Каролины, более половины библиотек, используемых Android-разработчиками для интеграции рекламы в приложения, имеют серьезные проблемы с безопасностью.

В исследовании было задействовано около ста тысяч приложений с Google Play, среди которых было выявлено порядка 100 наиболее часто используемых «рекламных» библиотек. Пристальное изучение этих элементов выявило их потенциальную опасность, которая проявлялась в загрузке непроверенного программного кода из сети, передаче пользовательских данных на удаленные сервера и других подозрительных действиях.

Специалисты отмечают, что большинство рекламных алгоритмов собирают данные о пользователе для создания целевой рекламы, но в большинстве случаев сведений о пользователе собирается чересчур много. К примеру, считыванию могут подвергаться списки вызовов абонентов, сведения о закладках браузера и даже данные об установленных приложениях. Кроме того, обращает на себя внимание, что некоторые библиотеки используют незащищенные механизмы загрузки контента из сети, без какой-либо проверки загружаемого кода.

Вызывает подозрения и способ определения «рекламщиками» местоположения пользователя. К примеру, библиотека Sosceo изучает статистику звонков пользователя и загружает данные в сеть, Mobus изучает SMS-сообщения для определения центра обслуживания.

По сведениям исследовательского центра Juniper MTC вредоносного ПО для мобильных устройств в 2011 г. стало больше на 155% по сравнению с предыдущим годом. А в I квартале 2012 г. произошел взрывной скачок вредоносного ПО для мобильных платформ, в основном, для Android. Отчасти это можно связать с улучшением технологии обнаружения вредоносного кода (показать все, что скрыто), но нельзя отрицать и объективный рост количества самих вирусов под мобильные платформы.

Как распределяются вирусы по мобильным платформам? «Лаборатория Касперского» обнаружила около 38 тыс. вредоносных программ для мобильных платформ. Из них 92,74% приходятся на Android, 5,86% — на J2ME, 1,18% — на Symbian и 0,22% — на остальные платформы (включая Windows Mobile, Blackberry и iOS)».

Только за последние несколько месяцев рост угроз для ОС Android составил 65%. При этом 30% вредоносных программ для Android распространяется через официальный интернет-магазин приложений Google Play. Среди злонамеренного ПО, проникающего на мобильные устройства, зафиксировано 37% троянских программ, передающихся с помощью SMS- и MMS-сообщений, а также 60% вирусов, способных устанавливать удаленный контроль над мобильным устройством.

Почему именно платформа Android так привлекает создателей вирусов? Можно выделить три основных причины.

Во-первых, Android — операционная система, которая применяется в устройствах разных производителей. Она пластична, удобна, и с каждым годом количество устройств на ней растет. В 2012 году по данным исследователей в мире насчитывается 500 миллионов пользователей этой мобильной платформы. Разумеется, такая массовость не может не привлечь злоумышленников.

Во-вторых, Android позволяет загружать приложения из сторонних источников. Наиболее популярные источники приложений – официальные магазины Google Play, Amazon App Store и Samsung Apps. Но ничего не мешает пользователю скачать приложение из любого источника или даже установить пришедшее по почте. Это открывает большой простор для мошеннических действий.

В-третьих, Google только недавно ужесточил правила и контроль за центром приложений Google Play (Andriod market). До этого (и по сей день) там можно было встретить массу вредоносных приложений.

Производители антивирусов давно в курсе проблем мобильной платформы Android. Под нее выпущено множество как платных, так и бесплатных антивирусных продуктов. Как правило, это мобильные версии от признанных борцов с компьютерными вирусами. Платные продукты, например, Dr.Web Light, Kaspersky Mobile Security Lite, бесплатные – avast! Mobile Security, «AVG Бесплатный антивирус».

К сожалению, антивирусные программы не защитят пользователя в двух случая. Как известно, разработчики вирусов не присылают образцы своей «продукции» разработчикам антивирусов. Поэтому антивирусное ПО всегда на шаг отстает от объекта борьбы. Шанс «словить» свеженаписанный вирус невелик, но есть.

Во втором случае речь пойдет об угрозе кошельку пользователя Android не от вируса, а от предприимчивых разработчиков, которые научились опустошать баланс мобильного пользователя Android, использующего телефон для выхода в интернет.

Сайты-ловушки существовали и до расцвета мобильного интернета. Еще свежи в памяти аудионаркотики, тесты, результаты которых Вы якобы получите, отправив SMS. Новые разновидности такого интернет-развода по-прежнему регулярно встречаются, становясь все более тонкими и изощренными. В последнее время часто встречаются поддельные сайты по поиску работы.

Пользователь, выходящий в интернет с мобильного телефона, особенно интересен злоумышленнику. Телефонное мошенничество переживает эпоху расцвета. Случайный клик на ссылку — и пользователь автоматически подписан на рассылку прогноза погоды, стоимость которой может варьироваться в зависимости от уровня запросов мошенников. Стоит отметить, что на этом все не останавливается.  Сначала операторы мобильной связи принудили сайты с платным контентом размещать условия оказания услуги. Но далеко не все дочитывают до конца страницы, где «серым по серому» и мелким шрифтом написано, сколько стоит услуга.

Услуги такого рода предоставляются с помощью так называемых «коротких номеров». Стоимость SMS на такие номера задается компанией, предоставляющей услуги по ним. Поэтому следующий шаг, на который операторов связи толкнула волна недовольства обманутых пользователей, стало упорядочивание пользования короткими номерами и информирования о стоимости таких услуг.

К сожалению, это решает проблему только при использовании стационарного компьютера. При посещении мошеннического сайта с мобильного телефона все предупреждения и информирование о стоимости услуг заботливо скрываются злоумышленниками под разноцветными яркими баннерами. Кроме того, новая технология позволяет автоматически определять номер телефона пользователя, и его уже даже не нужно вводить для того, чтобы лишиться энной суммы на счете.

100% защиты от этого пока не существует. Меры борьбы здесь скорее организационные — не ходить по сомнительным ссылкам, не устанавливать приложения из непроверенных источников. Отключить возможность ухода в «минус» и не иметь чрезмерно большого количества денег на счете. Регулярно проверять подписки своего телефона и отключать ненужные.

Таким образом, можно выделить несколько типов опасных объектов

1. Вредоносные файлы – целью таких приложений является выполнение заложенного в них скрытого функционала, незаметно для пользователя.
2. Потенциально опасные приложения – могут получить доступ к личным данным пользователя и применить полученную информацию по своему усмотрению, при этом напрямую или косвенно информируя о своей деятельности.

Так же следует отметить что Антивирусные компании знают не о всех опасных объектах существующих в реальности.

## 1.8 Структура Android приложений

Следует начать с того что все приложения для ОС Android распространяются в виде инсталляционных пакетов – файлов с расширением APK.

APK (Android Package) — формат архивных исполняемых файлов-приложений для [Android](http://ru.wikipedia.org/wiki/Android).

Каждое приложение Android [скомпилировано](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [упаковано](http://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в один файл, который включает в себя весь код приложения ([.DEX](http://ru.wikipedia.org/w/index.php?title=.DEX&action=edit&redlink=1) файлы), ресурсы, активы и файл [manifest](http://ru.wikipedia.org/w/index.php?title=.manifest&action=edit&redlink=1). Файл приложения может иметь любое имя, но расширение должно быть .APK. Например: myAppFile.apk.

Файлы с данным расширением хранятся в магазине [Google Play](http://ru.wikipedia.org/wiki/Google_Play), и загружаются с его помощью в [смартфон](http://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD) для их использования, либо устанавливаются пользователем вручную на устройстве.

Файлы этого формата не шифруются, являются подмножеством формата архива [ZIP](http://ru.wikipedia.org/wiki/ZIP).

Каждый .APK файл — это сжатый архив для исполнения в [DalvikVM](http://ru.wikipedia.org/wiki/Dalvik_virtual_machine) ([виртуальная машина](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0)), который может быть установлен не только на операционной системе Android.

APK файл как архив обычно содержит следующие директории:

* **META-INF :**

MANIFEST.MF: манифест файл

CERT.RSA: сертификат приложения

CERT.SF: список ресурсов и их SHA1 хеш-сумма например на Рисунке 6:

* Signature-Version: 1.0
* Created-By: 1.0 (Android)
* SHA1-Digest-Manifest: wxqnEAI0UA5nO5QJ8CGMwjkGGWE=
* ...
* Name: res/layout/exchange\_component\_back\_bottom.xml
* SHA1-Digest: eACjMjESj7Zkf0cBFTZ0nqWrt7w=
* ...
* Name: res/drawable-hdpi/icon.png
* SHA1-Digest: DGEqylP8W0n0iV/ZzBx3MW0WGCA=

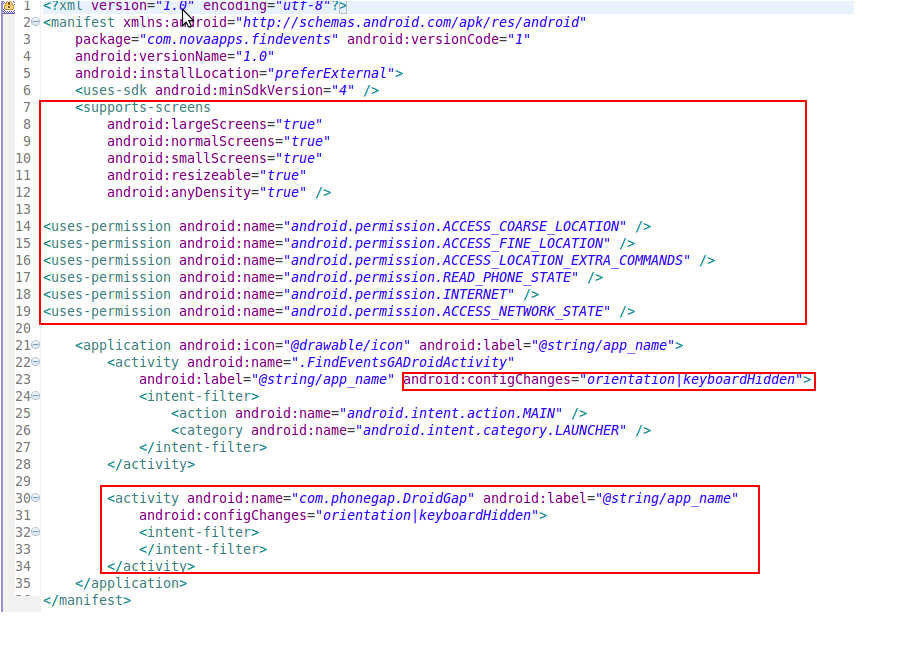
**Рисунок 6. Структура файла со списком ресурсов и их хеш-сумм.**

* **Директория lib:** содержит скомпилированный исполняемый код адаптированный под различные типы процессоров, обычно разделена на следующие директории:
  + armeabi: код только для ARM процессоров
  + armeabi-v7a: код для для процессоров ARMv7 и ниже только.
  + x86: скомпилированный код только для архитектуры x86
  + mips: скомпилированный код только для архитектуры MIPS
* **Директория res:** директория содержит файлы ресурсы не вошедшие в файл resources.arsc(см. ниже)
* **Директория assets:** содержит активы которые могут получены с помощью AssetManager
* **Файл AndroidManifest.xml:** дополнительный манифест файл, описывающий версию приложения, разрешения, используемые библиотеки. Как правило это файл идёт в формате binary XML, это формат файлов можно привести к читаемому виду с помощью сторонних утилит таких как AXMLPrinter2, apktool, или Androguard.
* **Файл classes.dex:** исполняемый файл виртуальной машины Dalvik, полученный путём преобразования скомпилированных JAVA классов с помощью утилиты DX. Утилита входит в состав Android SDK.
* **Файл resources.arsc:** файл содержит пре-компилированные ресурсы, например в виде бинарных XML файлов.

Из всего выше обозначенного в данной работе при анализе уровня опасности будут использоваться только два файла это AndroidManifest.xml и classes.dex. Ни их структуре остановимся более подробно.

**AndroidManifest.xml**

Данный файл, как уже говорилось ранее, содержит информацию о приложении, в том числе список требуемых разрешений приложения. В том числе на основе этих данных можно ранжировать уровень опасности приложения. Обратимся как его структуре: в первую очередь необходимо отметить что в установочном пакете androidmanifest.xml имеет бинарный вид, то есть преобразованный xml, хотя в оригинальном состоянии этот файл имеет структуру, обозначенную на Рисунке 7:



**Рисунок 7. Подтверждение установки приложения из стороннего источника.**

Следует отметить, что нас особенно интересуют поля обозначенные <uses-permission android:name=””/>. Эти полня показывают системе какие разрешения хочет получить приложения для своей работы. На их основе можно выставлять уровень опасности приложению.

В Таблице 1 приведены некоторые разрешения которые представляют наибольшую опасность :

|  |  |
| --- | --- |
| [ACCESS\_COARSE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION) | Приложение сможет получить доступ к приблизительному местоположению, полученному из сетевого расположения источников, таких как вышки сотовой связи и Wi-Fi. |
| [ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION) | Приложение сможет получить доступ к точному местоположению от места расположения источников, таких как GPS, вышек сотовой связи и Wi-Fi. |
| [CALL\_PHONE](http://developer.android.com/reference/android/Manifest.permission.html#CALL_PHONE) | Приложение сможет инициировать телефонный звонок, минуя пользовательский интерфейс Dialer для пользователя. |
| [CAMERA](http://developer.android.com/reference/android/Manifest.permission.html#CAMERA) | Приложение сможет сделать снимок встроенной камерой |
| [DELETE\_PACKAGES](http://developer.android.com/reference/android/Manifest.permission.html#DELETE_PACKAGES) | Позволяет приложению удалять пакеты. |
| [DEVICE\_POWER](http://developer.android.com/reference/android/Manifest.permission.html#DEVICE_POWER) | Позволяет приложению отключить питание устройства |

**Таблица 1. Описание некоторых опасных разрешений в ОС Android.**

Как видно из описаний некоторые разрешения можно группировать, выставив при этом уровень опасности целой группе функций, например разрешения [READ\_SMS](http://developer.android.com/reference/android/Manifest.permission.html#READ_SMS), [BRICK](http://developer.android.com/reference/android/Manifest.permission.html#BRICK) являются очень опасными и им можно присвоить уровень опасности 10(максимальный). Это вызвано тем, что под разрешением READ\_SMS понимается чтение личных данных пользователей, что является потенциально опасным для пользователя действием со стороны приложения. Под BRICK понимается отключение устройства в целом – что тоже очень опасно для пользователя потому что устройство полностью прекращает свою работу.

Исходя из всего выше сказанного, для ранжирования уровня опасности необходимо проанализировать все разрешения, выставить им уровни, так же можно группировать разрешения в составные при этом выставив общий уровень опасности приложению.

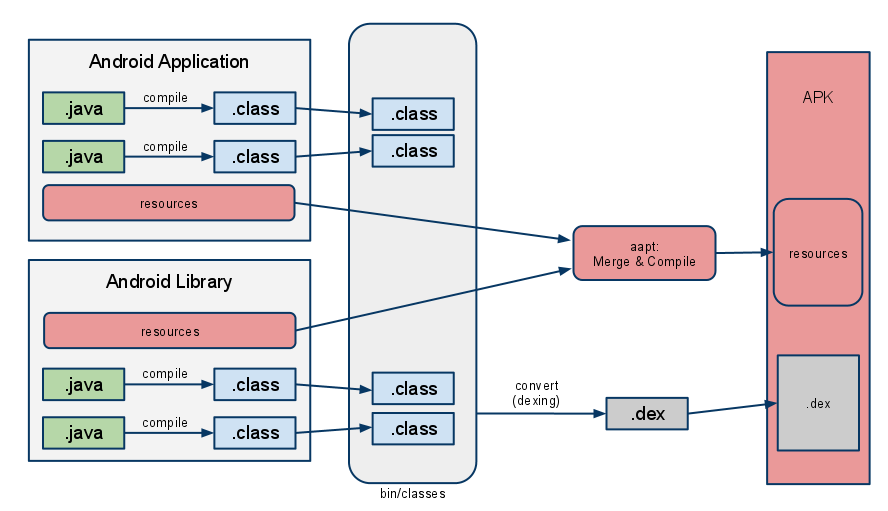
**classes.dex**

Данный файл носит в себе основной функционал приложения, содержит байт-код, понятный виртуальной машине Dalvik. Имеет следующую внутреннюю структуру, представленную в Таблице 2:

|  |  |
| --- | --- |
| header | Заголовок файла |
| string\_ids | Список идентификаторов строк, эти идентификаторы содержат информацию о всех строка в файле, независимо от их типа. |
| type\_ids | Список индетификаторов типов |
| proto\_ids | Список идентификаторов прототипов |
| field\_ids | Список всех идентификаторов полей, используемых в данном файле. |
| method\_ids | Список всех идентификаторов всех методов. |
| class\_defs | Список определений классов |
| data | Секция данных |
| link\_data | Информация о статически слинкованных файлах. |

**Таблица 2. Структура Dex файла.**

На самом деле это файл, содержащий в себе программный код для виртуальной машины Dalvik. Приложения для Android пишутся на языке Java, но после компиляции кода в .class-файлы, вызывается утилита dx, которая транслирует их в один файл classes.dex, являющийся основной составляющей APK файла. Общий алгоритм формирования dex представлен на Рисунке 8.

  
**Рисунок 8. Механизм формирования файла classes.dex.**

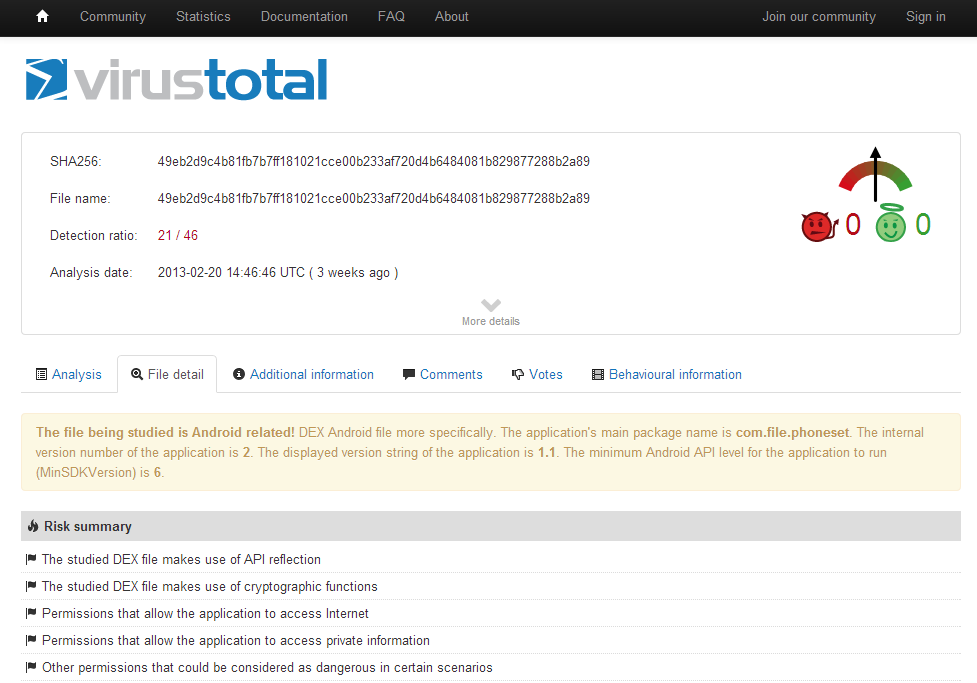
Следует отметить, что функционирование данного файла абсолютно связано с использованием API операционной системы. При этом исходный код приложения написан на объектно-ориентированном языке программирования JAVA. И является своего рода компиляцией в компиляции. Как следствие состоит из большего числа строк, содержащих имена методов API, имена различных констант. Вся эта информация может явным образом служить для понимания функционала приложения и как следствие ранжирования его уровня опасности, опираясь на использованные наборы API функций.

Так нужно понимать, что файл имеет четкую устоявшуюся структуру, которая позволяет получить из файла путём статического анализа необходимую информацию для понимания функционала. Для этого необходимо разобрать формат файла и научится извлекать строки, имена методов.

## 1.9 Анализ существующих систем определения уровня опасности.

В силу того что рынок решений безопасности в области Android находится на начальных стадиях развития ещё не существует системы, которая наглядно показала бы уровень опасности Android приложений. Антивирусные решения ограничиваются только одним уровнем – либо опасно либо нет. При этом в некоторых случаях пользователю не очевидна причина, по которой Антивирус счел тот или иной объект опасным.

Существуют системы, которые способны продемонстрировать уровень опасности. Одна из них сервис VirusTotal.com.



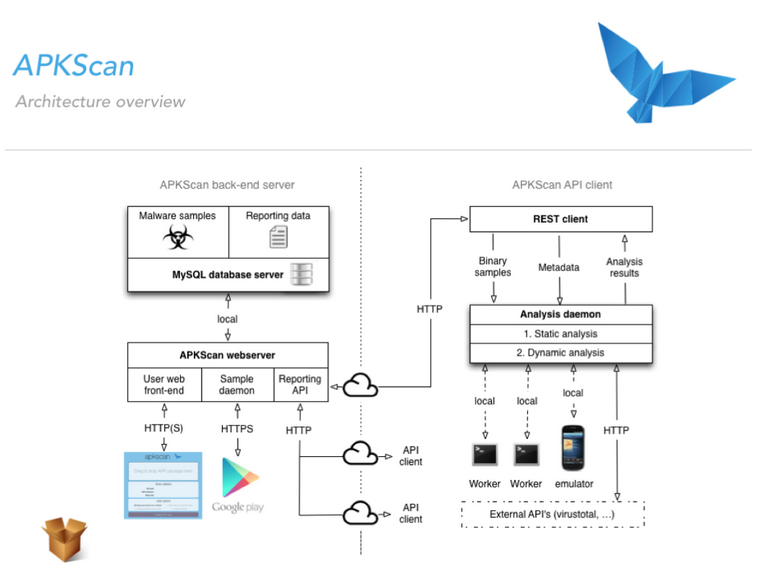
**Рисунок 9. Система VIRUSTOTAL, способная ранжировать уровень опасности.**

Основная функция которого – предоставление вердикта о детектировании загружаемого объекта многочисленными Антивирусными компаниями. Так же с недавнего времени ресурс начал предоставлять информацию по структуре установочного пакета Android приложения, детальную информацию по метаданным пакета, а так же результаты выполнения всего приложения в песочнице.

Всего этого более чем достаточно чтобы выставить уровень опасности по любому Android приложению. Имея при этом огромный багаж вредоносных приложений virustotal.com мог бы осуществлять ранжирование уровня опасности. Но по каким то причинам этого не делает.

Данный ресурс ограничивается сложным для пользователя «ИТОГО», из которого не ясно способно ли приложение причинить ущерб пользователю.

Существует так же другая система - APKSCAN



**Рисунок 9. Архитектура системы APKSCAN.**

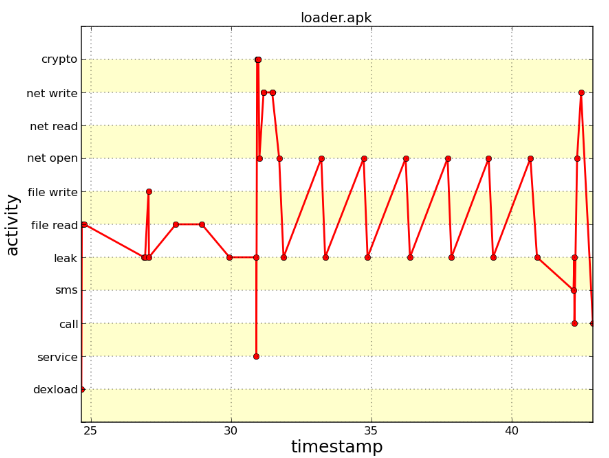
Данная система схожа с VIRUSTOTAL. Отличия заключаются в том, что у данной системы есть разного рода интерфейсы вовне. Например, эта системы может предоставлять результаты анализа приложения непосредственно в магазин приложений Google Play, который, на основе этих данных, может предупреждать пользователя об уровне опасности.

При этом наблюдается та же проблема что и с Virustotal: APKSCAN не хочет выступать центром экспертизы по уровням опасности причиной тому может быть их бизнес модель или же простое отсутствие системы ранжирования.

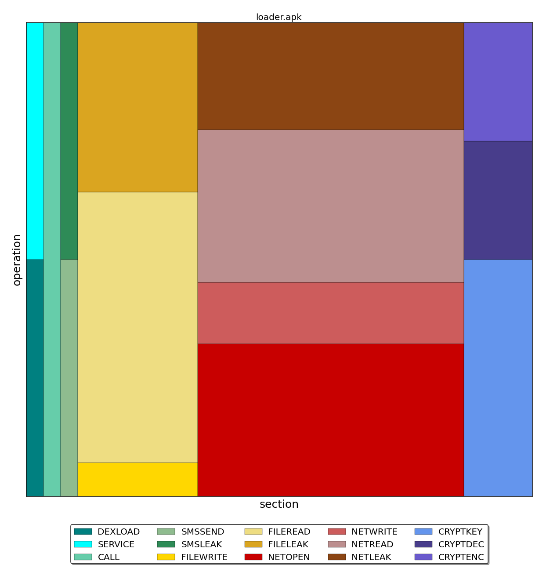
Так же существуют отдельно стоящие песочницы, результатом работы которых является список действий, совершенных приложением во время своей работы. В силу того что такие системы обособлены от «живых» примеров вредоносных приложений они используются только как части систем.

Пример таких песочниц

**Droidbox** (<https://code.google.com/p/droidbox/>)



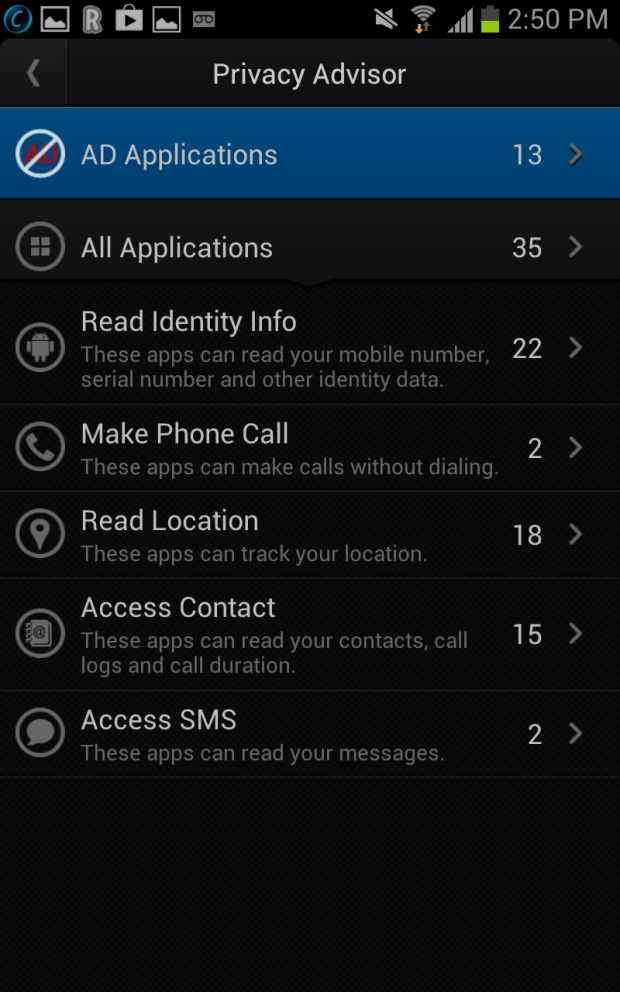
**Рисунок 10. Работа приложения в песочнице DROIDBOX.**



**Рисунок 11. Ранжирование уровней опасности в песочнице.**

В данном случае видно, что событию NETOPEN сопоставлено красное свечение квадрата. Это говорит о том, что создатели попытались ранжировать уровень опасности конкретного события, но не конечного значения уровня. Нет описания того что же сулит событие NETOPEN.

Так же существует модуль к антивирусному решению компании LOOKOUT под названием PRIVACY ADVISOR. Его логика работы основана только на статическом анализе AndroidManifest.xml и в целом не даёт репрезентативной картины опасности того или иного установленного на устройстве приложения. Но логика работы данного компонента наиболее проста и правдива с точки зрения анализа приложения. Пример работы компонента представлен на Рисунке 12:



**Рисунок 12. Lookout Privacy Advisor – пример работы.**

Все перечисленные системы обладают общим недостатком, он выражается в том, что данные системы не беспокоят пользователя. Он не вводят его в курс дела о грозящей опасности. Он лишь удаляют исключительно вредоносные приложения такие как например Trojan-SMS. В алгоритм работы таких приложений не заложен какой либо полезный функционал.

Исходя из вышесказанного, в общей схеме обеспечения безопасности ОС Android не хватает интерактивной работы с пользователем. Доведения до пользователя всех рисков связанных с установкой приложения. Особенно это касается случаев когда приложения устанавливаться из сторонних источников.

## 1.10 Выводы.

Операционная система Android нуждается в дополнительной защите. Защиту можно обеспечить различными путями:

* Использовать антивирусное ПО
* Внимательно просматривать требуемые разрешения, при установке приложений (что обычно игнорируется)
* Использовать только официальные источники приложений(то же не панацея так как и на официальных источниках есть опасные объекты)
* Воспользоваться сторонними разработками, такими как в данном дипломном проекте.

В заключение следует отметить очевидные недостатки существующих программных продуктов:

* Ни один из программных продуктов не отражает действительной картины уровня опасности, ограниваясь при этом только перечислением потенциально опасных функций.
* Не достаточно только статического анализа AndroidManifest.xml необходимо также заглянуть внутрь самого исполняемого файла приложения. Таким образом достигается полнота картины уровня опасности.
* Пользователем, устанавливающим приложения при помощи компьютера необходимо средства для анализа приложений.

# Раздел 2. Проектирование системы.

## 2.1 Анализ и выбор языка программирования для проектирования Системы

На сегодняшний день среди высоко уровневых языков программирования с объектно-ориентированной моделью проектирования, используемых для разработки программного обеспечения можно выделить следующие: С++, С#, Java. Рассмотрим каждый из них подробнее.

Язык программирования С++

C++ – универсальный компилируемый  язык программирования. Данный язык пригоден для любого круга задач, начиная от простых приложений пользовательского уровня и заканчивая системными компонентами, драйверами.

Но следует учитывать, что язык C++ предполагает некоторые требования к разработке ПО. Основные требования: необходимость следить за памятью, выделяемыми ресурсами, закрытию сессий. Если все ограничения соблюдены то, программное обеспечение, полученное в результате проектирования на этом языке будет максимально оптимизированным с точки зрения производительности.

Язык программирования С#

C# (си-шарп) – объектно-ориентированный язык программирования, разработанный корпорацией Microsoft как один из языков для работы с библиотекой .NET Framework. Удобство языка заключается в том, что типы данных, встроенных в язык являются классами. Это позволяет создавать сложные типы данных, таким образом упрощая написание кода. Сами библиотека .NET Framework обладает колоссальными возможностями и полностью исключает избыточный код. Благодаря тому, что конструкция языка предполагает полный контроль над ресурсами системы нет необходимости в контроле за памятью и указателях. Таким образом написание программы – чрезвычайно простой процесс как правило не требующий больших ресурсов. Неоспоримым преимуществом языка является интегрированная среда разработки Visual Studio. Которая так же способствует детальной проработке кода, а так обеспечивает сильнейшую проверку кода на ошибки.

Язык программирования Java

Java является языком программирования общего назначения, который специально разработан, чтобы иметь как можно меньше зависимостей от имплементаций. Он позволяет разработчикам приложений один раз написать приложение. В данном случае речь идёт о полной кроссплатформенности полученного кода. Java-приложения, как правило, скомпилированные в байт-код (файл class), который может работать на виртуальной машине Java независимо от архитектуры системы. По состоянию на 2012 год, Java является одним из самых популярных языков программирования, особенно для WEB-приложений. Java первоначально был разработан Джеймсом Гослинг в Sun Microsystems (которая затем объединилась с Oracle) и выпущен в 1995 году. Язык Java использовался в качестве одного из основных языков программирования в Sun Microsystems долгие годы. Большую часть синтаксиса Java получил от С и C++ поэтому разработчики не испытывали особых проблем при миграции на этот язык программирования

Java - объектно-ориентированный язык программирования поэтому он считается сравнительно простым. Благодаря тому, что нет необходимости следить за памятью, нет указателей, нет многократного наследования, приложения создаются быстро.

Есть и недостатки – в первую очередь хочется отметить невысокую скорость выполнения при наличии тяжелых графических интерфейсов. Как следствие появляется необходимость в более производителном оборудовании.

Сравнительный анализ и выбор языка программирования

При разработке программного обеспечения, заявленного в теме дипломного проекта, разработчику придется иметь дело со статическими методами анализа файлов, обращениями к базам данных, в условиях операционной системы Microsoft Windows. Платформа Java способствует быстрому написанию кода, однако, время выполнения основных операций в программе, разработанной с помощью этого языка может стать недопустимо большим. Язык С# также предоставляет разработчику удобство разработки, в виду сильной объектно-ориентированной основы языка, за счет того что необходимые для реализации библиотеки уже встроены в библиотеки языка и позволяют наиболее быстро и плотно интегрировать части проектируемой системы. Язык С++, менее удобен для программиста, по причине заметного увеличения времени написания программы, однако полученное приложение с большой долей вероятности будет работать очень быстро. Таким образом в данном дипломном проекте разработка будет вестись на языке C#.

## 2.2 Разработка Системы ранжирования уровней опасности Android приложений

## 2.2.1 Программные средства разработки

При разработке и отладке Системы, заявленной в теме дипломного проекта, потребуется задействовать следующие программные средства:

* Операционная система – Microsoft Windows 7;
* Среда разработки – Microsoft Visual Studio 2010 Express;

## 2.2.2 Аппаратная среда разработки

При разработке ПО Системы, заявленной в теме дипломного проекта, были задействованы следующие рабочая станция следующей конфигурации:

ПЭВМ типа IBM PC;

процессор: Intel Core i7 CPU 2630QM 2 ГГц;

Оперативная память: 6 Гб;

Жесткий диск: 600 Гб.

Сетевой адаптер – Realtek RTL8169/8110 Family Gigabit Ethernet;

Клавиатура, Мышь.

## 2.2.3 Алгоритм функционирования Системы

На Рисунке 13 представлена блок-схема алгоритма функционирования разрабатываемой системы.

Разрабатываемое программное обеспечение представляют собой самостоятельный исполняемый модуль – Обработчик Android приложений, а так же Базу данных, в которой хранятся данные о уже обработанных приложениях.

Программная реализация Обработчика включает в себя следующие логические модули.

Модуль обработки метаданных. Представляет собой программную реализацию обработки файла AndroidManifest.xml и ранжирования опасности приложения по этому файлу. Основные функции модуля обработки метаданных см. в Приложении 2.

Модуль обработки исполняемого кода. Представляет собой программную реализацию обработки файла classes.dex и ранжирования опасности приложения по этому файлу. Основные функции модуля обработки исполняемого кода см. в Приложении 2.

Модуль взаимодействия с базой данных. Представляет собой программную реализацию организации взаимодействия между Обработчиком и Базой данных. Основные функции модуля взаимодействия см. в Приложении 2.

Модуль распаковки. Представляет собой программную реализацию механизма распаковки архивов типа ZIP. Данный модуль необходим для извлечения данных из установочного файла APK. Основные функции модуля распаковки см. в Приложении 2.

Модуль взаимодействия с пользователем. Представляет собой программную реализацию организации взаимодействия между пользователем и внутренними данными Системы. Модуль существует только для отображения результатов работы системы пользователю.

.



**Рисунок 13. Блок-схема функционирования системы**

## 2.2.4 Выбор производителя и типа СУБД. Проектирование структуры базы данных.

Для реализации Системы требуется организовать хранение данных о тех приложениях, которые уже были обработаны в системе. Это необходимо для того чтобы обеспечить оперативное предоставление результатов ранжирования в том случае если запись запрашиваемого приложения уже содержится в системе.

Для того чтобы наглядно показывать пользователю информацию необходимо хранить описания всех разрешений с их описаниями, название пакета, ранжированный уровень опасности пакета полученный от анализа метаданных пакета, ранжированный уровень опасности пакета полученный от анализа бинарного исполняемого модуля приложения.

Таки образом структурная схема базы данных будет выглядеть, как показано на Рисунке 14.

На Рисунке 14 отображены таблицы со следующими именами:

* APP\_Ranking – основная таблицы соответствия ранжированного уровня опасности приложения MD5 хеш-сумме от установочного файла.
* APP\_Permissions – таблица соответствие конкретных разрешений используемых в приложении к MD5 хеш-сумме от приложения.
* Perm\_descr – таблица соответствия идентификатора (порядковый номер разрешения в шестнадцатеричном виде) разрешения его описанию, которое будет предоставлено пользователю.
* ExeMethods – список соответствия найденных опасных методов к MD5 хеш-сумме от приложения
* Method\_descr – таблица соответствия идентификатора опасного метода его описанию, которое будет предоставлено пользователю



**Рисунок 14. Структура таблиц баз данных Системы**

На ранке существуют множество различных реляционных СУБД, но наиболее популярные из них три: MySQL, Microsoft SQL Server, Oracle. В виду того что освоение ORACLE требует много времени, как правило связано с проектированием больших и сложных БД, использовать такие мощности для достаточно простой системы нецелесообразно.

СУБД MySQL отличается большой популярность, стабильностью и простой развёртывания. Но обладает слабой интегрированностью со средой разработки Microsoft Visual Studio. Так как в данном проекте используется язык программирования С# разумнее всего использовать СУБД Microsoft SQL Server. Даная система отличается высокой степенью интеграции со средой Visual Studio и предполагает плавный переход от решения простых задач к проектированию сложных Баз данных.

## 2.2.5 Механизм ранжирования приложений по метаданным и по исполняемому коду.

**Метаданные.**

В качестве метаданных в данном проекте будет использоваться только файл AndroidManifest.XML. Структура этого файла уже была описана в разделе в параграфе 1.6 раздела 1.

В этом файле наибольший интерес представляет список названий требуемых приложением разрешений (доступ в Интернет, отправка SMS и т.п.).

Исходя из общего списка разрешений, опубликованного на официальной странице операционной системы, проведём анализ опасности каждого разрешения и выставим ему уровень опасность путём попарного сравнения каждого разрешения.

**Исполняемый файл**

Для того чтобы понять уровень опасности по исполняемому файлу необходимо разобраться в его структуре. Решение задачи находится на поверхности – структура файла стандартизована разработчиками ОС. В данную структуру входит специальная секция в файле содержащая имена используемых в приложении имён методов и классов.

Для корректного ранжирования опасности по исполняемому файлу потребуется обратится к документации разработчика ОС составив список библиотечных классов и методов (примером метода является опасный SendTextMessage, класса SmsManager).

Результатом анализа библиотечных методов на предмет опасности может стать список опасных методов полученных в результате аналогичной схеме по Метаданным.

Таким образом имеем два списка с рангами:

* Список с разрешениями.
* Список с названиями методов.

В обоих случаях шкала опасности определяется Таблицей 3:

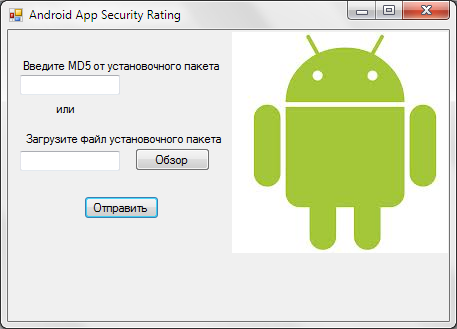
|  |  |
| --- | --- |
| Суммарный ранг | Уровень опасности |
| 0-4 | Приложение не опасно |
| 5-10 | Приложение потенциально опасно |
| 10- | Приложение крайне опасно |

**Таблица 3. Шкала опасности Android приложений.**

## 2.2.6 Интерфейс пользователя.

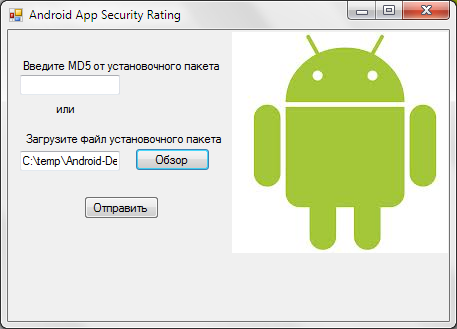
Разработанная система имеет наиболее простой интерфейс пользователя, построенный на генерации форм Microsoft Windows. Динамическая обработка данных пользователя и вывод результатов осуществляется при запросов к базе и отображении отчетов в отдельной форме.

Общий вид. На Рисунке 15 представлено основная форма Системы.



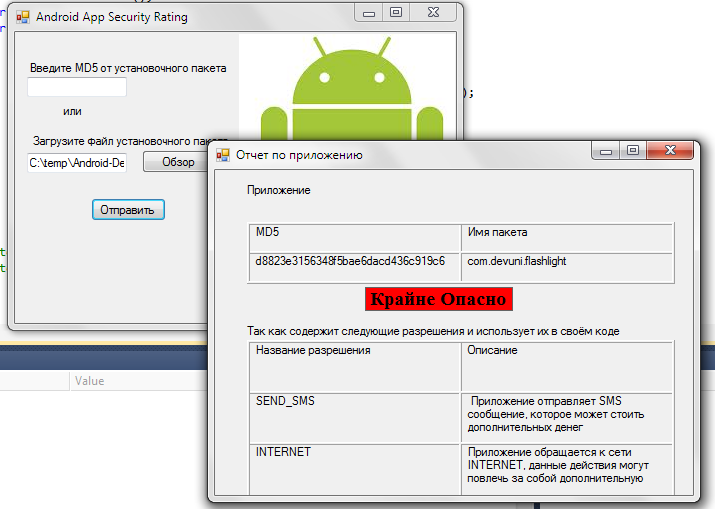
**Рисунок 14. Структура таблиц баз данных Системы**

Задание входных данных для обработки системой. На рисунке 15 представлен ввод входных данных.



**Рисунок 15. Ввод данных в систему**

Получение результатов обработки приложения. На рисунке 16 представлен ввод результата обработки приложения.



**Рисунок 16. Вывод результатов обработки приложения.**

Следует отметить, что в рамках разворачивания инфраструктуры непосредственный обработчик установочных пакетов может является консольным приложением, запускаемым при необходимости. Обработчик осуществляет статический анализ установочного пакета, заносит информацию в базу данных и даёт сигнал обработчику графики о том, что результаты готовы для показа пользователю.

## 2.2.7 Схема работы модулей системы

Следует отметить, что проектируемая система достаточно сложна, чтобы весь её функционал реализовать в одном блоке. Для проектирования такого рода программных средств целесообразно использовать модульный подход, при котором базовые операции Системы раздроблены на составляющие систему модули (блоки).

При таком подходе проектирования обеспечивается информативный вид кода, понятная логическая структура программного продукта. Проста поддержки Системы и её сопровождения.

На рисунке 17 представлена блок-схема взаимодействия модулей программных разрабатываемой системы.



**Рисунок 17. Блок-схема модулей системы.**

## 2.3 Выводы

В рамках проектирования системы в данной дипломной работе потребовалось изучить особенности популярных языков программирования, выявить их слабые и сильные стороны и принять решение о том в какой среде разрабатывать систему. Выбор пал на язык программирования C# и его библиотеку .Net.

Исходя из требований к данному дипломному проекту, был разработан алгоритм функционирования системы, предусматривающий использование базы данных.

Так как система предполагает взаимодействие с пользователем возникла необходимость разработать интерфейс взаимодействия. Для этого необходимо спроектировать оконные формы ввода данных и получения результатов.

Первичные запуски системы показали её эффективность при определении уровня опасности. Таким образом, можно сказать, что поставленные перед системой задачи выполнены.

Для дальнейшего внедрения системы требуется детальное тестирование.

# Раздел 3. Оценка эффективности принятых решений

## 3.1 Выбор и обоснование оценок эффективности.

В данном дипломном проекте разрабатывается программный продукт, как и любой другой продукт, он нуждается в оценке его эффектности. Так как в данном дипломном проекте к программному продукту предъявлены определённые технические требования, то необходимо оценить способность системы к решению поставленных перед ней задач. Данную оценку эффективности можно дать двумя способами.

1. Провести многократное тестирование системы «по белому ящику»
2. Провести однократное тестирование системы по черному ящику но используя заведомо подготовленные известные данные.

Наряду с функционалом системы необходимо оценить эффективность системы при сравнении ей с работой систем-конкурентов, данного рода оценку можно дать, только проверив временную характеристику работы систем на специально подготовленных тестовых данных.

Таким образом, при оценке эффективности имеет смысл провести полное тестирование, оценив то число по скольким приложениям система сможет выдать верный вердикт, а так же замерить скорость выполнения запросов. И сравнив скорость дать оценку эффективности.

## 3.2 Выбор и обоснование методики тестирования

Для осуществления тестирования были предложены следующие методы.

Так как тестирование проводится по белому ящику необходимо

1. Подготовить специальное тестовое приложение.
2. Загрузить приложение в систему
3. Посмотреть результаты работы.

Зная уровень опасности созданного приложения необходимо сверить результаты с эталоном.

Так же в целях тестирования необходимо проверить работу системы в исключительных ситуациях.

Проверка обработки исключений, возникающих в процессе работы Системы осуществляется путем создания критических условий, характерных для нестабильного функционирования испытуемой системы, и по её реакции принимается решение о правильности обработки исключительных ситуаций..

Ещё одним вариантом методики является проверка хаотично выбранного приложения доступного из сети INTERNET. В данном случае методика предполагает следующие шаги:

1. Найти ресурс сети INTERNET, распространяющий приложения для ОС Android. Скачать несколько приложений
2. Вручную обработать приложения силами таксировщика выявив опасный функционал (если есть)
3. Загрузить файлы в систему.
4. Сверить результаты с анализом.

Если сравнивать две методики между собой, то разумнее всего использовать подготовленные заранее данные. Это проще с точки зрения подготовки данные для теста. Быстрее с точки зрения анализа результатов. Но есть и минус – такое тестирование слегка отдалено от действительности, но в целом весьма эффективное.

Таким образом при тестирование будет использоваться методика с заготовленными данными.

Так же надо учесть методики тестирования на корректную обработку системой критических ситуации. Для этого потребуется так же специальное приложение, структура которого специально испорчена.

Исходя из вышесказанного, тестирование будет по трём методиками:

1. Загрузка в систему специального тестового приложения – корректного.
2. Загрузка в систему специального тестового приложения – некорректного.
3. Загрузка в систему любого приложения – замер времени выполнения.

## 3.3 Тестирование разработанного программного обеспечения

На данном этапе проектирования системы необходимо провести оценочные мероприятия по выявлению тех или иных особенностей и неполадок в работе системы.

Необходимо провести тестирование системы. Тестирование разработанного ПО можно проводить двумя путями.

Первый вариант – это тестирование по черному ящику. В рамках этого тестирования на вход системе подаются данные, вердикт по которым известен заранее. Система тестируется комплексно, главным образом результаты работы.

Второй вариант – это анализ работы тестовых заданий, времени обработки результата, анализ надежности и удобства.

Так как тестирование по черному ящику предполагает сильное вмешательство в код системы возрастает вероятность человеческой ошибки. Поэтому в данном дипломном проекте будет проводится тестирование только по белому ящику.

Объектом тестирования является Система автоматического ранжирования уровня опасности Android приложений.

Цель тестирования - Испытание разработанной Системы автоматического ранжирования уровня опасности Android приложений для проверки соответствия требованиям технического задания и оценки надежности, эффективности.

## 3.3.1 Требования к Системе

В техническом задании на дипломный проект заявлены следующие требования к разрабатываемой Системе.

**Общие требования**

Данная Система ранжирования уровней опасности Android приложения должна выполнять следующие основные функции:

* создание отчетов в формате WEB страницы(вывод данных)
* статический анализ приложения
* хранение информации в формате Баз данных
* обработка загружаемых файлов (ввод данных)
* хранение признаков уровней опасности в формате БД(ввод данных)
* обработка сохраненной информации;

*Входные данные:* На вход системе подаётся установочный пакет приложения для ОС Android. Данный пакет имеет специфическую структуру: формат пакета – архив типа ZIP, состоит как минимум из двух файлов: classes.dex и AndroidManifest.xml.

Если при работе системы в неё будет загружен установочный пакет иного формата, то система должна сообщить об ошибке.

Формат файла classes.dex бинарный, для его обработки необходимо разработать специальный статический анализатор.

*Выходные данные:*  Результатом работы системы должен файл отчета содержащий следующие поля:

* Название анализируемого файла
* Имя установочного пакета
* Хеш-сумма MD5, посчитанная от установочного пакета
* Числовое значение уровня опасности приложения с обозначенной рядом шкалой и словесным описанием уровня.
* Список словестных описаний полученного уровня.

Данные так же дублируются на дисплей пользователя в виде отдельного окна интерфейса Системы.

**Требования к надежности**

Для того чтобы Система работала корректно в течении долгого времени необходимо спроектировать в системе следующие функции:

° проверку входных данных на предмет повреждения и соответствие требованиям;

° ведение диалога с пользователем – сообщая ему об ошибке;

° возможность повторной загрузки.

**Требования к условиям эксплуатации**

Для работы Системы абсолютно необходимо наличие библиотеки .Net Framework.

**Требования к составу и параметрам технических средств**

Для функционирования программного обеспечения необходимо наличие следующих технических средств:

° IBM РС-совместимая машина;

° монитор (для ввода-вывода информации);

**Требования к информационной и программной совместимости**

Данная программа должна представлять собой самостоятельный исполняемый модуль.

Система должна быть реализована с использованием одного из известных языков программирования (Java, C#) и работать под управлением операционной системы Microsoft Windows.

**Требования к программной документации**

Дня разрабатываемого программного обеспечения должна быть разработана следующая техническая документация:

° текст программы; (см. Приложение 2);

° программа и методика тестирования; (см. раздел 3.3);

° руководство пользователя. (см. Приложение 1)

## 3.3.2 Средства тестирования

Для тестирования разрабатываемой системы необходимо использовать 1 ПЭВМ со следующими основными характеристиками:

Процессор – Intel Core i7 CPU 2630QM 2 ГГц;

Оперативная память – 6 Гб;

Жесткий диск – 600 Гб;

Сетевой адаптер – Realtek RTL8169/8110 Family Gigabit Ethernet;

Клавиатура, Мышь;

Операционная система – Microsoft Windows 7;.

Тестирование разработанной Системы выполнялось согласно методике испытаний, указанной в разделе 3.2.

Результаты тестирования приведены в таблице 3.

## 3.3.3 Результаты тестирования системы

|  |  |  |
| --- | --- | --- |
| № пп раздела 3.2 | Метод тестирования | Результат  тестирования |
| 3.2.1 п.1 | Проверить, что загруженное в систему тестовое приложение обработалось системой корректно | Успех |
| 3.2.1 п.2 | Проверить, что загруженное в систему повреждённое тестовое приложение обработалось системой корректно | Исключительные ситуации обрабатываются корректно |
| 3.2.1 п.3 | Замер скорости выполнения запроса на хаотичном приложении | 1 минута |

**Таблица 3. Результаты тестирования Системы.**

В данном тестировании рассматривалась эффективность разработанной системы (временная оценка), по сравнению с некоторыми существующими на сегодняшний день программными средствами определения уровня опасности Android приложений.

В данном тестировании использовались следующие программные продукты: ресурс virustotal.com, Песочница DroidBox, Ативирусное ПО LOOKOUT

Суть сравнительного тестирования состоит в оценке времени за которое вышеуказанные программные средства определят уровень опасности Android приложения в сравнении в разработанной в данной дипломной работе Системой.

|  |
| --- |
| Наименование программного продукта |
| Спец файл 1 |
| virustotal.com | 1 минута |
| DroidBox | 5 минут |
| LOOKOUT | 10 минут |
| Система ранжирования уровня опасности Android приложений | 1 минута |

**Таблица 4. Результаты сравнительного тестирования.**

## 3.4 Оценка и результатов

Результаты тестирования показывают, что разработанная в данной дипломной работе Система ранжирования уровней опасности Android отвечает предъявленным к ней требованиям. Выполняет заложенный в неё функционал.

Для проведения оценок было собрано достаточно данных:

Исходя из таблицы 3, можно сделать вывод о том, что что разработанная в рамках данной дипломной работы Система эффективна с точки зрения заложенного в ней функционала и предъявленных к ней требований.

Исходя из таблицы 4, можно сделать вывод о том, что разработанная в рамках данной дипломной работы Система, превосходит по временным характеристикам возможности представленных в сравнительном тестировании программные средства. Таким образом Система доказывает свою эффективности в разрезе временных характеристик работы.

## 3.7 Надежность разрабатываемой системы.

Одними из самых серьезных проблем программного обеспечения (ПО) являются дороговизна, низкая надежность и свойство системы сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях эксплуатации. Чем качественнее система, тем она надежнее, и наоборот. Многие специалисты считают первый из этих недостатков продолжением второго. Поскольку программное обеспечение по самой своей природе ненадежно, его тестирование и сопровождение требует постоянных и существенных расходов. Необнаруженные ошибки в программах могут явиться причиной отказов, последствия которых могут оказаться существенными.

Следует начать с того что озвучить термины надёжности для этого необходимо обратится к трудам "Надёжность программного обеспечения” Майерса и к "Надёжность программных средств" Липаева:

* Если программное обеспечение не выполняет ожидаемых от него функций, то такая ситуация показывает наличие ошибки в ПО;
* отказ программного обеспечения – это появление в нем ошибки;
* под надёжностью программного обеспечения следует понимать вероятность безотказной работы в течении заранее заданного промежутка времени, полученной с учетом потерь полученных пользователем в результате каждого отдельного взятого отказа.

Исходя из данных терминов можно сделать вывод:

* Надежность программного обеспечения является не только внутренним свойством программы;
* надежность ПО – это функция как самого ПО, так и ожиданий (действий) его пользователей.

Основными причинами ошибок ПО являются:

* большая сложность ПО (например, по сравнению с аппаратурой ЭВМ);
* неправильный перевод информации из одного представления в другое на макро и микро уровнях.

Сложность системы является одной из главных причин низкой надежности ПО. В общем случае, сложность объекта является функцией взаимодействия (количества связей) между его компонентами. В борьбе со сложностью ПО используются две концепции. Иерархическая структура. Иерархия позволяет разбить систему по уровням понимания (абстракции, управления). Концепция уровней позволяет анализировать систему, скрывая несущественные для данного уровня детали реализации других уровней. Иерархия позволяет понимать, проектировать и описывать сложные системы.

Источниками ошибок (угрозами надежности) ПО являются:

* внутренние – ошибки проектирования, ошибки алгоритмизации, ошибки программирования, недостаточное качество защиты, ошибки в документации;
* внешние – ошибки пользователей, сбои и отказы аппаратуры ЭВМ, искажение информации в каналах связи, изменения конфигурации системы.

Важным этапом жизненного цикла ПО, определяющим качество и надежность системы, является тестирование. Тестирование – процесс выполнения программ с намерением найти ошибки.

Этапы тестирования:

* автономное тестирование – контроль отдельного программного модуля отдельно от других модулей системы;
* тестирование функций – контроль выполнения системой автоматизируемых функций;
* комплексное тестирование – проверка соответствия системы требованиям пользователей;
* тестирование полноты и корректности документации – выполнение программы в строгом соответствии с инструкциями;
* тестирование конфигураций – проверка каждого конкретного варианта поставки (установки) системы.

Существуют две стратегии при проектировании тестов:

* тестирование по отношению к спецификациям (документации), не заботясь о тексте программы;
* тестирование по отношению к тексту программы, не заботясь о спецификациях.

Разумный компромисс лежит где-то посередине, смещаясь в ту или иную сторону в зависимости от функций, выполняемых конкретным модулем, комплексом или подсистемой.

Показателем надежности программного комплекса может служить вероятность отсутствия обнаружения программных ошибок в течение определенного промежутка времени при эксплуатации программы в расчетном режиме в информационной системе.

Безотказность – свойство объекта непрерывно сохранять работоспособность в течение некоторой наработки или в течение некоторого времени.

Среднее время наработки на ошибку:

, где

 - интенсивность ошибок ПО;

Интенсивность ошибок разрабатываемого ПО рассчитывается по формулам:

,

, где

 – фактическое время отладки;

 – коэффициент крутизны линии, характеризующий скорость роста надежности;

 – число обнаруженных ошибок за время отладки t;

 – общее число строк;

 – коэффициент, учитывающий влияние методологии программирования на надежность ПО;

 – коэффициент, учитывающий использование i-той технологии программирования;

 – коэффициент, учитывающий использование i-того языка программирования;

 – коэффициент, учитывающий использование i-той платформы программирования;

В данном ПО использовались:

– объектно-ориентированная технологий программирования ();

– среда разработки Visual Studio ();

– операционная система семейства Windows ().



Количество строк кода .

Отладка производилась тестированием ПО в течение 140 часов.

Результаты приведены в таблице 5 .

|  |  |  |
| --- | --- | --- |
| Количество  ошибок | Продолжительность отладки t, часы | Интенсивность ошибок , |
| 32 | 20 | 0.00060 |
| 24 | 20 | 0.00045 |
| 18 | 20 | 0.00033 |
| 13 | 20 | 0.00024 |
| 10 | 20 | 0.00018 |
| 7 | 20 | 0.00013 |
| 5 | 20 | 0.00009 |

**Таблица 5. Интенсивность ошибок.**

Интенсивность ошибок, как было сказано выше, рассчитываем по формуле:



Остальные значения считаются аналогично.

На основе полученных данных построим кривую зависимости интенсивности ошибок от времени отладки (рисунок 18).



**Рисунок 18.Зависимость интенсивности ошибок от времени отладки.**

Как показано выше, функциональная зависимость интенсивности ошибок ПО от времени отладки описывается экспоненциальным законом и зависит от коэффициента крутизны линии, характеризующей скорость роста надежности , и от фактического времени отладки ПО. Анализ результатов тестирования ПО позволил определить .

Таким образом, интенсивность ошибок разрабатываемого ПО составляет:

,

.

Для разрабатываемого ПО средняя наработка на ошибку составит:



Вероятность безошибочной работы системы в течение времени t рассчитывается по формуле:

.

Пусть t = 8 часам. Тогда:



Ремонтопригодность – свойство объекта быть приспособленным к предупреждению и обнаружению отказов и повреждений, к восстановлению работоспособности и исправности в процессе технического обслуживания и ремонта.

Для ПО ремонтопригодность характеризуется коэффициентом готовности. Готовность – свойство ПО быть в состоянии выполнять требуемую функцию в данный момент времени при заданных условиях использования.

Коэффициент готовности рассчитывается по формуле:

 , где

 – средняя наработка на ошибку (6943 часа);

 – время восстановления программы (30 минут = 0.5 часа).

Таким образом, коэффициент готовности разрабатываемой системы равен:



## 3.8 Выводы

1. Разработанная в рамках данной дипломной работы Система удовлетворяет всем требованиям, описанным настоящей пояснительно записки.
2. Разработанная в рамках данной дипломной работы Система успешно прошла тестирование по всем пунктам методики испытаний приведенной в разделе 3.2 пояснительно записки.
3. Использование специальных подготовленных баз данных оправдало себя и дало хорошие показатели производительности.
4. Для повышения надежности и минимизации времени на восстановления работоспособности после сбоев рекомендуется делать резервные копии базы данных на внешний носитель.

# Заключение

В данном дипломном проекте ставилась задача разработки Системы автоматического ранжирования уровня опасности Android приложений.

Необходимость в такой системе на моём предприятии была вызвана тем, что нам на обработку присылают большие объемы разного рода приложений, и мы не в силах их обработать вручную. Для того чтобы справиться с потоком был необходим инструмент, позволяющий пре-анализировать приложения. Отличить потенциально опасное приложение от чистого. Для этого не было подходящих готовых решений на рынке. Исходя из вышесказанного можно утверждать, что разработка такой системы является чрезвычайно актуальной для предприятия.

В ходе дипломного проектирования были решены следующие задачи.

На этапе исследований предметной области была разобрана структура операционной системы Android в части её безопасности. Так же была разобрана структура приложений для данной операционной системы. Были получены навыки обратной инженерии. Оценены возможности автоматической обработки приложений – их автоанализ. Для сокращения временных затрат на повторную обработку было принято решение ввести в систему СУБД – для хранения данных.

На этапе разработки Системы, были выбраны средства разработки, а именно язык программирования C#, было принято решение, что интерфейс Системы должен быть в виде WEB страницы. Таким образом достигнута независимость системы от типа клиента, его операционной системы, браузера. Была выбрана СУБД MS SQL Server в качестве хранения данных.

На этапах разработки и тестирования был разработан интерфейс взаимодействия с пользователем, который и является ядром системы. Так же была создана методика тестирования всей системы. Была разработана соответствующая документация. Документация включила в себя наглядное пособие по развёртыванию системы и её эксплуатации с иллюстрациями. На ряду с программной частью была проделана работа по оценке принятый в проекте решений а так же проведен расчет надёжности системы.

В экономическом разделе проекте было приведено технико-экономическое обоснование разработки Системы, а так же была проиллюстрирована экономическая эффективность создаваемой Системы.

В разделе по охране труда было проведено исследование опасных и вредных факторов при работе с ПЭВМ, методы защиты от них, а также приведены эргономические требования к рабочим местам.

Результатом работы над дипломным проектом является разработанная Система ранжирования уровня опасности Android приложений, которая позволила начать автоматическую обработку более 2000 файлов в сутки, поступающих на анализ в Лабораторию Касперского. Таким образом аналитики Лаборатории смогли оперативно реагировать на поступающие запросы и выявлять новые неизвестные опасные объекты.

# Список литературы

"C# 4.0. Полное руководство ", Герберт Шилдт, Вильямс, 2013 г.

"Язык программирования C# 5.0 и платформа .NET 4.5", Эндрю Троелсен, Вильямс, 2013 г.

"Инфраструктура программных проектов. Соглашения, идиомы и шаблоны для многократно используемых библиотек .NET", Кржиштоф Цвалина, Брэд Абрамс, Вильямс, 2011 г.

"PHP. Объекты, шаблоны и методики программирования", Мэтт Зандстра, Вильямс, 2011 г.

"Microsoft SQL Server 2008. Реализация и обслуживание" ,Майк Хотек, Русская Редакция, 2011 г.

"SQL и реляционная теория. Как грамотно писать код на SQL", К. Дж. Дейт, Символ-Плюс, 2010 г.

"Базы данных", И. А. Кумскова, КноРус, 2011 г.

"Безопасность глобальных сетевых технологий", Владимир Зима, Александр Молдовян, Николай Молдовян, BHV-Санкт-Петербург, 2003 г.

"Анонимность и безопасность в Интернете", Денис Колисниченко, 2012 г

"Основы информационной безопасности. Краткий курс", Феникс, 2008 г.

"Надёжность программного обеспечения” Майерс" Г. /Мир. - М., 1980. - 360 с.

"Надёжность программных средств" Липаев В.В. /СИНТЕГ. - М., 1998. - 232 с.

Интернет ресурсы:

http://www.kaspersky.ru

http://virustotal.com

https://code.google.com/p/droidbox/

http://habrahabr.ru

https://source.android.com/tech/security/

http://play.google.com/

http://www.computerra.ru

Трудовой кодекс Российской Федерации.

ГОСТ 12.0.003-99 Опасные и вредные производственные факторы. Классификация.

ГОСТ 12.1.030-81 (2001) Электробезопасность. Защитное заземление. Зануление.

ГОСТ ССБТ 12.1.124-83 Средства защиты от статического электричества.

СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным ЭВМ и организации работы.

ГОСТ 12.1.019-96 Система стандартов безопасности труда (ССБТ). Общие требования и номенклатура видов защиты.

Санитарные правила и нормы (СанПиН) 2.2.1/2.1.1.1278-03 Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий.

ГОСТ 12.1.004-91 Пожарная безопасность. Общие требования.

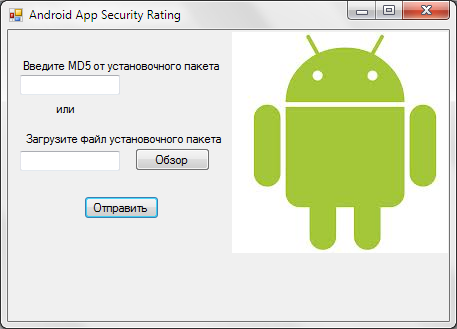
## Приложение 1. Руководство пользователя

Данная система предназначена для ранжирования уровня опасности Android приложений. Состоит из двух основных частей: основная часть – файл с именем AppRatings.exe, написанный на языке С# с использованием .NET фреймворка и база данных на основе СУБД Microsoft SQL Server 2013,

Таким образом для нормального функционирования системы потребуется провести подготовительные работы следующего характера:

1. Установить Microsoft SQL Сервер, создать внутри СУБД пользователя, выполнить файл db.sql, текст которого содержится в Приложении 2. Дать доступ на созданные базы ранее созданному пользователю с уровнем доступа RW. Ссылка на дистрибутив <http://care.dlservice.microsoft.com//dl/download/5/6/5/5656F917-B0AE-415B-9136-A9DC7915143A/SQLServer2012SP1-FullSlipstream-RUS-x64.iso?lcid=1049&cprod=sql2012rtm32cta>
2. При необходимости установить комплект библиотек .Net FrameWork

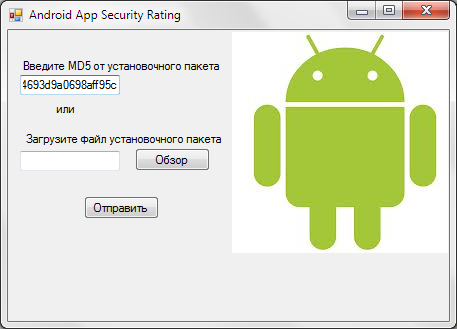
В результате всех описанных выше действий при запуске файла AppRatings.exe должно быть успешное отображение интерфейса системы представленное на Рисунке 1



**Рисунок 1. Успешно работающий интерфейс системы.**

**Работа в системе**

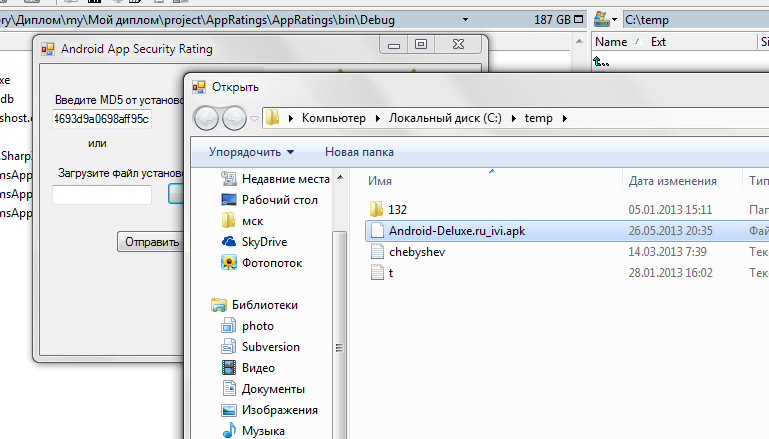
Работу в системе можно строить двумя путями

1. Если у пользователя нет исследуемого файла а есть лишь его хеш-сумма, то необходимо ввести её в верхнее поле ввода как показано на рисунке 2. После нажатия на кнопку «Отправить» система сделать запрос к базе данных, и выдаст результат о том известно ли ей информация по указанной хеш-сумме. 

**Рисунок 2. Ввод данных в систему.**

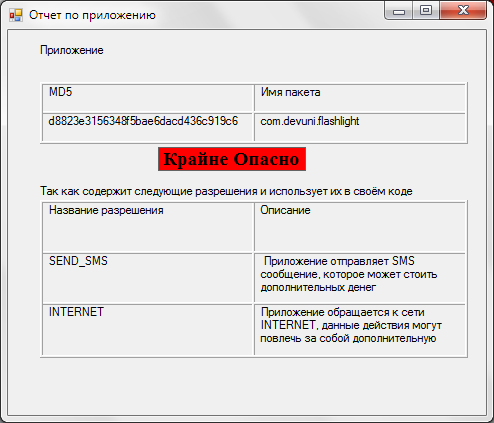
1. Если у пользователя есть исследуемый

Для того чтобы обработать файл необходимо нажать на кнопку «ОБЗОР» и выбрать интересующий файл как показано на рисунке 3



**Рисунок 3. Загрузка файла в систему.**

После того как выбор сделан необходимо нажать на кнопку отправить – система после непродолжительного ожидания выдаст результаты, пример которых показан на рисунке 3



**Рисунок 3. Результаты работы системы**

## Приложение 2. Распечатка текстов программных модулей

Модуль взаимодействия с пользователем:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Form2 fr2 = new Form2(this);

string md5 = textBox1.Text.ToString();

string File\_Path = textBox2.Text.ToString();

if (md5.Length == 0 & File\_Path.Length == 0)

{

MessageBox.Show("Пожалуйста, заполните хотя бы одну форму ввода");

}

if (md5.Length != 0 & File\_Path.Length != 0)

{

MessageBox.Show("За один раз можно создать только один запрос");

}

if (md5.Length > 0 || File\_Path.Length > 0) {

fr2.ShowDialog();

}

}

private void button2\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

textBox2.Text = openFileDialog.FileName;

}

}

}

}

Модуль вызова обработчика данных и вывода информации пользователю:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Xml;

using System.IO;

using System.Data.SqlClient;

using System.Data;

namespace WindowsFormsApplication1

{

public partial class Form2 : Form

{

private Form1 form1;

public Form2(Form1 form1)

{

// TODO: Complete member initialization

InitializeComponent();

this.form1 = form1;

}

private void Form2\_Load(object sender, EventArgs e)

{

string File\_Path = form1.textBox2.Text;

string File\_md5 = form1.textBox1.Text;

string App\_Perms = "";

int rank = 0;

if (File\_Path.Length != 0){

Manifest.readManifest(File.ReadAllBytes(File\_Path));

var permissions = Manifest.permissions;

var app\_name = Manifest.AppName;

XmlDocument doc = new XmlDocument();

try

{

doc.LoadXml(System.IO.File.ReadAllText("permissions.xml"));

}

catch (Exception ex) { MessageBox.Show("Не могу открыть файл permissions.xml!");}

XmlNodeList items = doc.GetElementsByTagName("Permission");

foreach (XmlNode x in items) {

foreach (string p in permissions)

{

if (p.Contains(x.Attributes[0].Value)) {

App\_Perms += x.Attributes[3].Value;

try

{

if (Convert.ToInt16(x.Attributes[1].Value) > 5)

{

string hoho = p.Substring(p.LastIndexOf(".") + 1, p.Length - p.LastIndexOf(".") - 1);

listBox1.Items.Add(hoho);

// App\_Perms += x.Attributes[3].Value;

}

}

catch (ArgumentOutOfRangeException ex) { MessageBox.Show("No"); }

rank += Convert.ToInt32(x.Attributes[1].Value);

}

}

}

label4.Text = tools.GetMD5(File\_Path);

label5.Text = app\_name;

if (rank > 10) {

label6.BackColor = System.Drawing.Color.Red;

label6.Text = "Крайне опасно";

label7.Text += " содержит следующие опасные методы:";

this.listBox1.Show();

}

if (rank < 10 & rank > 5)

{

label6.BackColor = System.Drawing.Color.Gray;

label6.Text = "Потенциально опасно";

label7.Text += " содержит следующие потенциально опасные методы:";

this.listBox1.Show();

}

if (rank <= 5)

{

label6.BackColor = System.Drawing.Color.Green;

label6.Text = "Безопасно";

label7.Text += " не содержит опасных методов";

}

string connStr = @"Data Source=(local)\SQLEXPRESS;

Initial Catalog=Android;

Integrated Security=True";

SqlConnection conn = new SqlConnection(connStr);

try

{

//пробуем подключится

conn.Open();

}

catch (SqlException se) { MessageBox.Show("Не могу соединится с сервером"); }

//Заносим данные в основную таблицу с уровнем опасности

string s\_prepare = "insert into App\_Ranking (file\_md5,App\_Name,metadrank,exerank) values (0x" + label4.Text + ",'" + app\_name + "'," + rank + ",0);";

SqlCommand sqlcmd = new SqlCommand(s\_prepare,conn);

try

{

sqlcmd.ExecuteNonQuery();

}

catch { MessageBox.Show("Не могу добавить запись в базу данных, скорее всего она уже существует"); }

//Заносим данные в дополнительную таблицу с разрешением по каждому приложению

string s\_prepare1 = "insert into App\_Permissions (file\_Md5,permissions) values (0x" + label4.Text + ",'" + App\_Perms + "');";

SqlCommand sqlcmd1 = new SqlCommand(s\_prepare1, conn);

try

{

sqlcmd1.ExecuteNonQuery();

}

catch { MessageBox.Show("Не могу добавить запись в базу данных, скорее всего она уже существует"); }

conn.Close();

conn.Dispose();

}

if (File\_md5.Length != 0)

{

string connStr = @"Data Source=(local)\SQLEXPRESS;

Initial Catalog=Android;

Integrated Security=True";

SqlConnection conn = new SqlConnection(connStr);

try

{

//пробуем подключится

conn.Open();

}

catch (SqlException se) { MessageBox.Show("Не могу соединится с сервером"); }

//Запрос к основной таблице уровня опасности, заполнение верхней таблицы

string s\_prepare = "select \* from App\_Ranking where file\_md5 = 0x" + File\_md5 + ";";

SqlCommand sqlcmd = new SqlCommand(s\_prepare, conn);

try

{

sqlcmd.ExecuteNonQuery();

}

catch { MessageBox.Show("Не могу выполнить запрос"); }

using (SqlDataReader dr = sqlcmd.ExecuteReader(CommandBehavior.CloseConnection))

{

while (dr.Read())

{

label4.Text = File\_md5;

label5.Text = dr.GetValue(1).ToString();

rank = Convert.ToInt32(dr.GetValue(2));

if (rank > 10)

{

label6.BackColor = System.Drawing.Color.Red;

label6.Text = "Крайне опасно";

label7.Text += " содержит следующие опасные методы:";

this.listBox1.Show();

}

if (rank < 10 & rank > 5)

{

label6.BackColor = System.Drawing.Color.Gray;

label6.Text = "Потенциально опасно";

label7.Text += " содержит следующие потенциально опасные методы:";

this.listBox1.Show();

}

if (rank <= 5)

{

label6.BackColor = System.Drawing.Color.Green;

label6.Text = "Безопасно";

label7.Text += " не содержит опасных методов";

}

}

}

//Запрос к основной таблице разрешений, заполнение нижней таблицы

SqlConnection conn1 = new SqlConnection(connStr);

try

{

//пробуем подключится

conn1.Open();

}

catch (SqlException se) { MessageBox.Show("Не могу соединится с сервером"); }

string s\_prepare1 = "select \* from App\_Permissions where file\_md5 = 0x" + File\_md5 + ";";

SqlCommand sqlcmd1 = new SqlCommand(s\_prepare1, conn1);

try

{

sqlcmd1.ExecuteNonQuery();

}

catch { MessageBox.Show("Не могу выполнить запрос"); }

XmlDocument doc = new XmlDocument();

try

{

doc.LoadXml(System.IO.File.ReadAllText("permissions.xml"));

}

catch (Exception ex) { MessageBox.Show("Не могу открыть файл permissions.xml!");}

XmlNodeList items = doc.GetElementsByTagName("Permission");

using (SqlDataReader dr = sqlcmd1.ExecuteReader(CommandBehavior.CloseConnection))

{

while (dr.Read())

{

App\_Perms = Convert.ToString(dr.GetValue(1));

string[] perms = new string[App\_Perms.Length / 2];

int k = 0;

int i = 0;

while (k < App\_Perms.Length/2)

{

while(i < App\_Perms.Length)

{

perms[k] += App\_Perms[i];

i++;

if (i % 2 == 0) { k++; break; }

}

}

foreach (string s in perms)

{

foreach (XmlNode x in items)

{

if (s == x.Attributes[3].Value)

{

if (Convert.ToInt16(x.Attributes[1].Value) > 5)

{

listBox1.Items.Add(x.Attributes[0].Value);

}

break;

}

}

}

}

}

conn.Close();

conn.Dispose();

}

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

XmlDocument doc = new XmlDocument();

try

{

doc.LoadXml(System.IO.File.ReadAllText("permissions.xml"));

}

catch (Exception ex) { MessageBox.Show("Не могу открыть файл permissions.xml!"); }

XmlNodeList items = doc.GetElementsByTagName("Permission");

foreach (XmlNode x in items) {

if (listBox1.SelectedItem.Equals(x.Attributes[0].Value)) {

label8.Text = x.Attributes[2].Value;

label8.Show();

}

}

}

}

}

Модуль обработки данных 1:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace WindowsFormsApplication1

{

class Dex

{

public Dex(string \_filename)

{

// Strings\_all = new List<string>();

FileFullName = \_filename;

}

public Dex(byte[] buff)

{

//Strings\_all = new List<string>();

this.buff = buff;

FileFullName = "";

}

public string FileFullName;

public byte[] buff;

private static byte[] DEXMAGIC = { 0x64, 0x65, 0x78, 0x0a, 0x30, 0x33, 0x35, 0x00 }; //dex\n035\0

private const uint HEADER\_SIZE = 0x70;

private const uint MAGIC\_SIZE = 8;

private const uint SHA\_SIZE = 0x14;

private const uint ENDIAN\_TAG = 0x12345678;

public string[] Strings\_all;

public string [] Methods;

public string[] Types;

public string[] Fields;

public class \_ImageDexHeader

{

public byte[] magic;

public UInt32 checksum;

public byte[] sha;

public UInt32 file\_length;

public UInt32 header\_lenght;

public UInt32 endiantag;

public UInt32 link\_size;

public UInt32 link\_off;

public UInt32 map\_off;

public UInt32 string\_ids\_size;

public UInt32 string\_ids\_off;

public UInt32 type\_ids\_size;

public UInt32 type\_ids\_off;

public UInt32 proto\_ids\_size;

public UInt32 proto\_ids\_off;

public UInt32 field\_ids\_size;

public UInt32 field\_ids\_off;

public UInt32 method\_ids\_size;

public UInt32 method\_ids\_off;

public UInt32 class\_defs\_size;

public UInt32 class\_defs\_off;

public UInt32 data\_size;

public UInt32 data\_off;

public bool Read(BinaryReader BR)

{

if (BR.BaseStream.Length < GetSize())

{

return false;

}

try

{

BR.BaseStream.Seek(0, SeekOrigin.Begin);

magic = BR.ReadBytes((int)MAGIC\_SIZE);

checksum = BR.ReadUInt32();

sha = BR.ReadBytes((int)SHA\_SIZE);

file\_length = BR.ReadUInt32();

header\_lenght = BR.ReadUInt32();

endiantag = BR.ReadUInt32();

link\_size = BR.ReadUInt32();

link\_off = BR.ReadUInt32();

map\_off = BR.ReadUInt32();

string\_ids\_size = BR.ReadUInt32();

string\_ids\_off = BR.ReadUInt32();

type\_ids\_size = BR.ReadUInt32();

type\_ids\_off = BR.ReadUInt32();

proto\_ids\_size = BR.ReadUInt32();

proto\_ids\_off = BR.ReadUInt32();

field\_ids\_size = BR.ReadUInt32();

field\_ids\_off = BR.ReadUInt32();

method\_ids\_size = BR.ReadUInt32();

method\_ids\_off = BR.ReadUInt32();

class\_defs\_size = BR.ReadUInt32();

class\_defs\_off = BR.ReadUInt32();

data\_size = BR.ReadUInt32();

data\_off = BR.ReadUInt32();

}

catch

{

return false;

}

for (int i = 0; i < MAGIC\_SIZE; i++)

if (magic[i] != DEXMAGIC[i]) return false;

//if (!magic.Equals(DEXMAGIC)) return false;

if (endiantag != ENDIAN\_TAG) return false;

if (file\_length != BR.BaseStream.Length) return false;

if (header\_lenght != HEADER\_SIZE) return false;

return true;

}

public UInt32 GetSize()

{

return HEADER\_SIZE;

}

}

private bool ReadStrings(BinaryReader BR, uint string\_ids\_offset, uint string\_ids\_size)

{

Strings\_all = new string[string\_ids\_size];

if (BR.BaseStream.Length <= string\_ids\_size \* 4 + string\_ids\_offset) return false;

BR.BaseStream.Position = string\_ids\_offset;

for (int i = 0; i < string\_ids\_size; i++)

{

BR.BaseStream.Position = string\_ids\_offset + i \* 4;

var str\_ids\_offset = BR.BaseStream.Position;

var str\_offset = BR.ReadUInt32();

BR.BaseStream.Position = str\_offset;

var str\_size = ReadLEB128(BR);

var buff = BR.ReadBytes((int)str\_size);

var name = "";

foreach (var b in buff)

{

name += (char)b;

}

Strings\_all[i] = name;

}

return true;

}

private bool ReadMethods(BinaryReader BR, uint method\_off, uint method\_size)

{

Methods = new string[method\_size];

if (BR.BaseStream.Length <= 8 \* method\_size + method\_off) return false;

BR.BaseStream.Position = method\_off;

for (int i = 0; i < method\_size; i++)

{

var class\_idx = BR.ReadUInt16();

var proto\_idx = BR.ReadUInt16();

var name\_idx = (int)BR.ReadUInt32();

string item = Strings\_all[name\_idx];

string cl\_name = Types[(int)class\_idx];

if (!cl\_name.EndsWith(";")) cl\_name += ";";

Methods[i] = (cl\_name + item);

// Console.WriteLine(cl\_name);

// Console.WriteLine(item);

// Console.WriteLine(Types[(int)class\_idx] + item);

}

return true;

}

private bool ReadFields(BinaryReader BR, uint field\_off, uint field\_size)

{

Fields = new string[field\_size];

if (BR.BaseStream.Length <= 8 \* field\_size + field\_off) return false;

BR.BaseStream.Position = field\_off;

for (int i = 0; i < field\_size; i++)

{

var class\_idx = BR.ReadUInt16();

var proto\_idx = BR.ReadUInt16();

var name\_idx = (int)BR.ReadUInt32();

string item = Strings\_all[name\_idx];

Fields[i] = (Types[(int)class\_idx] + item);

// Console.WriteLine(Types[(int)class\_idx] + item);

}

return true;

}

private bool ReadTypes(BinaryReader BR, uint types\_off, uint types\_size)

{

Types = new string[types\_size];

if (BR.BaseStream.Length <= 4 \* types\_size + types\_off) return false;

BR.BaseStream.Position = types\_off;

for (int i = 0; i < types\_size; i++)

{

var name\_idx = (int)BR.ReadUInt32();

string item = Strings\_all[name\_idx];

//if (!Types.Contains(item))

Types[i] = item;

}

return true;

}

public static UInt64 ReadLEB128(BinaryReader BR)

{

UInt64 res = 0;

int shift = 0;

while (true)

{

byte value = BR.ReadByte();

res |= (UInt64)((value & 0x7f) << shift);

shift += 7;

if ((value & 0x80) == 0) break;

}

return res;

}

private bool GetBinareReader(out BinaryReader br)

{

if (string.IsNullOrEmpty(FileFullName))

{

MemoryStream ms = new MemoryStream(buff);

br = new BinaryReader(ms);

return true;

}

br = new BinaryReader(File.OpenRead(FileFullName));

if (br.BaseStream.Length < 20) return false;

var h = br.ReadBytes(4);

//64 65 78 0A

if ((h[0] == 0x64) && (h[1] == 0x65) && (h[2] == 0x78) && (h[3] == 0x0a)) return true;

//50 4B 03 04

if ((h[0] == 0x50) && (h[1] == 0x4b) && (h[2] == 0x03) && (h[3] == 0x04))

{

br.BaseStream.Position = 0;

var dex = tools.GetDex(br.ReadBytes((int)br.BaseStream.Length));

if (dex.Length < 20) return false;

MemoryStream ms = new MemoryStream(dex);

br = new BinaryReader(ms);

return true;

}

return false;

}

public bool Read()

{

BinaryReader br;

if (!GetBinareReader(out br)) return false;

\_ImageDexHeader header = new \_ImageDexHeader();

if (!header.Read(br)) return false;

if (!ReadStrings(br, header.string\_ids\_off, header.string\_ids\_size)) return false;

if (!ReadTypes(br, header.type\_ids\_off, header.type\_ids\_size)) return false;

if (!ReadMethods(br, header.method\_ids\_off, header.method\_ids\_size)) return false;

if (!ReadFields(br, header.field\_ids\_off, header.field\_ids\_size)) return false;

return true;

}

public bool Read\_small()

{

BinaryReader br;

if (!GetBinareReader(out br)) return false;

\_ImageDexHeader header = new \_ImageDexHeader();

if (!header.Read(br)) return false;

if (!ReadStrings(br, header.string\_ids\_off, header.string\_ids\_size)) return false;

if (!ReadTypes(br, header.type\_ids\_off, header.type\_ids\_size)) return false;

if (!ReadMethods(br, header.method\_ids\_off,header.method\_ids\_size)) return false;

return true;

}

public List<string> Read\_strings()

{

BinaryReader br;

if (!GetBinareReader(out br)) return new List<string>();

\_ImageDexHeader header = new \_ImageDexHeader();

if (!header.Read(br)) return new List<string>();

if (!ReadStrings(br, header.string\_ids\_off, header.string\_ids\_size)) return new List<string>();

return Strings\_all.ToList();

}

}

}

Модуль обработки данных 2:

using ICSharpCode.SharpZipLib.Zip;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

namespace WindowsFormsApplication1

{

class Manifest

{

public static List<string> actions;

public static List<string> permissions;

public static string AppName;

public static void readManifest(byte[] apk)

{

actions = new List<string>();

permissions = new List<string>();

//List<string> res = new List<string>();

if ((apk[0] != 0x50) || (apk[1] != 0x4b) || (apk[2] != 0x03) || (apk[3] != 0x04))

//return "";

return;

try

{

MemoryStream ms = new MemoryStream(apk);

using (ZipInputStream zip = new ZipInputStream(ms))

{

ZipEntry theEntry;

while ((theEntry = zip.GetNextEntry()) != null)

{

if (theEntry.Name == String.Empty)

continue;

Console.WriteLine(theEntry.Name);

if (theEntry.Name.Contains("AndroidManifest.xml"))

{

MemoryStream rs = new MemoryStream();

int size = 2048;

int ll = 0;

byte[] data = new byte[2048];

while (true)

{

size = zip.Read(data, 0, data.Length);

if (size > 0)

{

ll += size;

rs.Write(data, 0, size);

}

else

{

break;

}

}

// Thread.Sleep()

if (rs.Length < 0x20) continue;

var unp\_buff = new byte[rs.Length];

rs.Position = 0;

rs.Read(unp\_buff, 0, unp\_buff.Length);

rs.Flush();

decompressXML(unp\_buff);

//return decompressXML(unp\_buff);

}

}

}

//no manifest

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

// return "";

}

// decompressXML -- Parse the 'compressed' binary form of Android XML docs

// such as for AndroidManifest.xml in .apk files

private static int endDocTag = 0x00100101;

private static int startTag = 0x00100102;

private static int endTag = 0x00100103;

private static string decompressXML(byte[] xml)

{

int tabs = 0;

int numbStrings = LEW(xml, 4 \* 4);

// StringIndexTable starts at offset 24x, an array of 32 bit LE offsets

// of the length/string data in the StringTable.

int sitOff = 0x24; // Offset of start of StringIndexTable

// StringTable, each string is represented with a 16 bit little endian

// character count, followed by that number of 16 bit (LE) (Unicode) chars.

int stOff = sitOff + numbStrings \* 4; // StringTable follows StrIndexTable

// XMLTags, The XML tag tree starts after some unknown content after the

// StringTable. There is some unknown data after the StringTable, scan

// forward from this point to the flag for the start of an XML start tag.

int xmlTagOff = LEW(xml, 3 \* 4); // Start from the offset in the 3rd word.

// Scan forward until we find the bytes: 0x02011000(x00100102 in normal int)

for (int ii = xmlTagOff; ii < xml.Length - 4; ii += 4)

{

if (LEW(xml, ii) == startTag)

{

xmlTagOff = ii; break;

}

}

int off = xmlTagOff;

int indent = 0;

int startTagLineNo = -2;

while (off < xml.Length)

{

int tag0 = LEW(xml, off);

//int tag1 = LEW(xml, off+1\*4);

int lineNo = LEW(xml, off + 2 \* 4);

//int tag3 = LEW(xml, off+3\*4);

int nameNsSi = LEW(xml, off + 4 \* 4);

int nameSi = LEW(xml, off + 5 \* 4);

if (tag0 == startTag)

{ // XML START TAG

int tag6 = LEW(xml, off + 6 \* 4); // Expected to be 14001400

int numbAttrs = LEW(xml, off + 7 \* 4); // Number of Attributes to follow

//int tag8 = LEW(xml, off+8\*4); // Expected to be 00000000

off += 9 \* 4; // Skip over 6+3 words of startTag data

String name = compXmlString(xml, sitOff, stOff, nameSi);

//tr.addSelect(name, null);

startTagLineNo = lineNo;

//Console.WriteLine(tag6.ToString() + "\t" + name);

// Look for the Attributes

tabs++;

StringBuilder sb = new StringBuilder();

for (int ii = 0; ii < numbAttrs; ii++)

{

int attrNameNsSi = LEW(xml, off); // AttrName Namespace Str Ind, or FFFFFFFF

int attrNameSi = LEW(xml, off + 1 \* 4); // AttrName String Index

int attrValueSi = LEW(xml, off + 2 \* 4); // AttrValue Str Ind, or FFFFFFFF

int attrFlags = LEW(xml, off + 3 \* 4);

int attrResId = LEW(xml, off + 4 \* 4); // AttrValue ResourceId or dup AttrValue StrInd

off += 5 \* 4; // Skip over the 5 words of an attribute

String attrName = compXmlString(xml, sitOff, stOff, attrNameSi);

String attrValue = attrValueSi != -1

? compXmlString(xml, sitOff, stOff, attrValueSi)

//: "resourceID 0x" + Integer.toHexString(attrResId);

: "resourceID 0x" + attrResId.ToString("X");

sb.Append(" " + attrName + "=\"" + attrValue + "\"");

//tr.add(attrName, attrValue);

}

if (name.ToLower() == "uses-permission")

{

string rr = sb.Replace("\"", "").ToString();

rr = rr.Substring(rr.IndexOf("name=") + "name=".Length);

//prtIndent(indent, rr);

permissions.Add( rr);

}

else if (name.ToLower() == "permission")

{

string rr = sb.Replace("\"", "").ToString();

rr = rr.Substring(rr.IndexOf("name=") + "name=".Length);

//prtIndent(indent, rr);

permissions.Add(rr);

}

else if (name.ToLower() == "action")

{

string rr = sb.Replace("\"", "").ToString();

rr = "action = " + rr.Substring(rr.IndexOf("name=") + "name=".Length);

actions.Add(rr);

// res.Add(rr);

}

indent++;

}

else if (tag0 == endTag)

{ // XML END TAG

indent--;

off += 6 \* 4; // Skip over 6 words of endTag data

String name = compXmlString(xml, sitOff, stOff, nameSi);

}

else if (tag0 == endDocTag)

{

break;

}

else

{

break;

}

}

);

string result = "";

return result;

} // end of decompressXML

private static String compXmlString(byte[] xml, int sitOff, int stOff, int strInd)

{

if (strInd < 0) return null;

int strOff = stOff + LEW(xml, sitOff + strInd \* 4);

return compXmlStringAt(xml, strOff);

}

private static String spaces = " ";

private static string prtIndent(int indent, String str)

{

Console.WriteLine(spaces.Substring(0, (int)Math.Min(indent \* 2, spaces.Length)) + str);

return spaces.Substring(0, (int)Math.Min(indent \* 2, spaces.Length)) + str;

}

// compXmlStringAt -- Return the string stored in StringTable format at

// offset strOff. This offset points to the 16 bit string length, which

// is followed by that number of 16 bit (Unicode) chars.

private static String compXmlStringAt(byte[] arr, int strOff)

{

int strLen = arr[strOff + 1] << 8 & 0xff00 | arr[strOff] & 0xff;

char[] chars = new char[strLen];

for (int ii = 0; ii < strLen; ii++)

{

chars[ii] = (char)arr[strOff + 2 + ii \* 2];

}

return new string(chars); // Hack, just use 8 byte chars

} // end of compXmlStringAt

// LEW -- Return value of a Little Endian 32 bit word from the byte array

// at offset off.

private static int LEW(byte[] arr, int off)

{

return (int)(arr[off + 3] << 24 & 0xff000000 | arr[off + 2] << 16 & 0xff0000

| arr[off + 1] << 8 & 0xff00 | arr[off] & 0xFF);

} // end of LEW

}

}

Модуль распаковки:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Security.Cryptography;

using System.Security.Cryptography.X509Certificates;

using ICSharpCode.SharpZipLib.BZip2;

using ICSharpCode.SharpZipLib.Zip;

using ICSharpCode.SharpZipLib.Zip.Compression;

using ICSharpCode.SharpZipLib.Zip.Compression.Streams;

using ICSharpCode.SharpZipLib.GZip;

namespace WindowsFormsApplication1

{

class tools

{

public static void unpack(string file)

{

FastZip fz = new FastZip();

string unpdir = file + "\_unp";

if (!Directory.Exists(unpdir))

Directory.CreateDirectory(unpdir);

fz.ExtractZip(file, unpdir, null);

}

public static string GetMD5(string filename)

{

try

{

FileStream file = new FileStream(filename, FileMode.Open);

MD5 md5 = new MD5CryptoServiceProvider();

byte[] retVal = md5.ComputeHash(file);

file.Close();

StringBuilder sb = new StringBuilder();

for (int i = 0; i < retVal.Length; i++)

{

sb.Append(retVal[i].ToString("X2"));

}

return sb.ToString();

}

catch

{

return "000000T000000000000000000000000";

}

}

public static byte[] GetMD5(byte[] buff)

{

MD5 md5 = new MD5CryptoServiceProvider();

byte[] retVal = md5.ComputeHash(buff);

return retVal;

}

public static string ConvertMD5(byte[] md5)

{

StringBuilder sb = new StringBuilder();

for (int i = 0; i < md5.Length; i++)

{

sb.Append(md5[i].ToString("X2"));

}

return sb.ToString();

}

public static byte[] ConvertMD5(string line)

{

line = line.Trim();

var bytes = new byte[16];

line = line.Replace("0x", "");

if (line.Length != 32) return bytes;

for (int i = 0; i < 16; i++)

{

string s = "" + line[i \* 2] + line[i \* 2 + 1];

int b = Convert.ToInt32(s, (int)16);

bytes[i] = (byte)b;

}

return bytes;

}

public static byte[] GetMD5b(string filename)

{

// FileStream file = new FileStream(filename, FileMode.Open);

BinaryReader br = new BinaryReader(File.OpenRead(filename));

var buff = br.ReadBytes((int)br.BaseStream.Length);

br.Close();

MD5 md5 = new MD5CryptoServiceProvider();

byte[] retVal = md5.ComputeHash(buff);

return retVal;

}

public static byte[] GetDex(byte[] apk)

{

try

{

MemoryStream ms = new MemoryStream(apk);

using (ZipInputStream zip = new ZipInputStream(ms))

{

ZipEntry theEntry;

while ((theEntry = zip.GetNextEntry()) != null)

{

if (theEntry.Name == String.Empty)

continue;

if (theEntry.Name!="classes.dex")continue;

MemoryStream rs = new MemoryStream();

int size = 2048;

int ll = 0;

byte[] data = new byte[2048];

while (true)

{

size = zip.Read(data, 0, data.Length);

if (size > 0)

{

ll += size;

rs.Write(data, 0, size);

}

else

{

break;

}

}

if (rs.Length < 0x20) continue;

var unp\_buff = new byte[rs.Length];

rs.Position = 0;

rs.Read(unp\_buff, 0, unp\_buff.Length);

rs.Flush();

if (unp\_buff[0] != 0x64) continue;

if (unp\_buff[1] != 0x65) continue;

if (unp\_buff[2] != 0x78) continue;

if (unp\_buff[3] != 0x0a) continue;

return unp\_buff;

}

}

return new byte[1];

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

return new byte[1];

}

}

}

}

SQL Сценарий для выполнения на Microsoft SQL Server

create database Android;

use Android;

create table App\_Ranking (file\_md5 binary(16) PRIMARY KEY, App\_Name varchar(25), metadrank int, exerank int);

create table App\_Permissions (file\_Md5 binary(16) FOREIGN KEY references app\_ranking(File\_md5), permissions varchar(10));

create table Perm\_Descr (id char(2), descr text);

create table ExeMethods (file\_Md5 binary(16) FOREIGN KEY references app\_ranking(File\_md5), method1 int, method2 int, method3 int, method4 int, method5 int);

create table Method\_Descr (id int, descr text);