ISSN 2500-316X (Online)

https://doi.org/10.32362/2500-316X-2019-7-6-44-55

УДК 004.451.9

Оценка уровня информационной безопасности мобильной операционной системы Android

Д.А. Изергин,
М.А. Еремеев,
Ш.Г. Магомедов@,
С.И. Смирнов

МИРЭА – Российский технологический университет, Москва 119454, Россия
@Автор для переписки, e-mail: izergind@gmail.com; magomedov_sh@mirea.ru

Одним из главных направлений развития информационных систем является повышение эффективности сбора, обработки и обмена информацией на основе внедрения современных технологий передачи данных, автоматизированного дистанционного мониторинга и управления. Краеугольным камнем данной концепции являются мобильные устройства, решающие вопрос оперативного обмена данными и их обработки. Современные мобильные сервисы, использующиеся в том числе для обмена и обработки персонализированных, банковских и критически важных данных, появляются вследствие неуклонного роста количества преступлений в сфере информационной безопасности в отношении и с использованием мобильных устройств. Широкое применение данных устройств для доступа к защищаемой информации, содержащейся в информационных системах, придало вопросу обеспечения информационной защищенности особое значение.

Предметом исследования данной статьи является оценка текущего состояния механизмов обеспечения информационной безопасности мобильных операционных систем, составляющих основу структуры эпизодических распределенных мобильных сетей, на примере ОС (операционной системы) Android. В статье рассмотрены проблемы развития мобильной экосистемы и методы, направленные на их решение, основные векторы вредоносного воздействия, способы противодействия средствам статического и динамического анализа и современные механизмы защиты.

Ключевые слова: мобильные операционные системы, Android, вредоносное воздействие, информационная безопасность, механизмы защиты.

Для цитирования: Изергин Д.А., Еремеев М.А., Магомедов Ш.Г., Смирнов С.И. Оценка уровня информационной безопасности мобильной операционной системы Android. *Российский технологический журнал.* 2019;7(6):44-55. https://doi.org/10.32362/2500-316X-2019-7-6-44-55

Information security evaluation for Android mobile operating system

Dmitriy A. Izergin[@],
Mikhail A. Eremeev,
Shamil G. Magomedov,
Stanislav I. Smirnov

MIREA – Russian Technological University, Moscow 119454, Russia
[@]Corresponding author, e-mail: izergind@gmail.com; magomedov_sh@mirea.ru

One of the main directions of information systems development is to increase the efficiency of collecting, processing and exchanging information through the introduction of modern data transfer technologies, automated remote monitoring and control. The cornerstone of this concept is mobile devices that solve the issue of operational data exchange and processing. Modern mobile services used including the exchange and processing of personalized, banking and critical data are the result of the steady increase in the number of crimes in the field of information security in relation to and using mobile devices. The widespread use of these devices for access to protected information contained in information systems has given special importance to the issue of information security.

The subject of this research is to assess the current state of information security mechanisms for mobile operating systems that form the basis of the structure of episodic distributed mobile networks. The Android OS (operating system) was used as an example. The article discusses the problems of the development of a mobile ecosystem and methods aimed at solving them, the main vectors of malicious impact, ways of countering the means of static and dynamic analysis and modern protection mechanisms.

Keywords: mobile operating systems, Android, malicious impact, information security, protection mechanisms.

For citation: Izergin D.A., Eremeev M.A., Magomedov Sh.G., Smirnov S.I. Information security evaluation for Android mobile operating system. *Rossiiskii tekhnologicheskii zhurnal* = Russian Technological Journal. 2019;7(6)44-55 (in Russ.). https://doi.org/10.32362/2500-316X-2019-7-6-44-55

Введение

Согласно отчетам рейтингового агентства Statcounter (рис. 1), на первый квартал 2019 года Android является наиболее распространенной операционной системой (ОС) в мире с результатом 38% от общего количества установленных ОС, что налагает обязанность данной ОС быть наиболее качественной в области информационной безопасности [1]. Однако, несмотря на то, что компания Google и производители мобильных устройств постоянно совершенствуют систему безопасности, но открытость исходного кода и обширная фрагментация платформы делает данную систему одной из самых уязвимых для вредоносного воздействия.

Текущее состояние безопасности Android

Главной причиной фрагментации экосистемы Android является применяемая технология при создании мобильных устройств, основанная на системе «кристалл/чип» (system

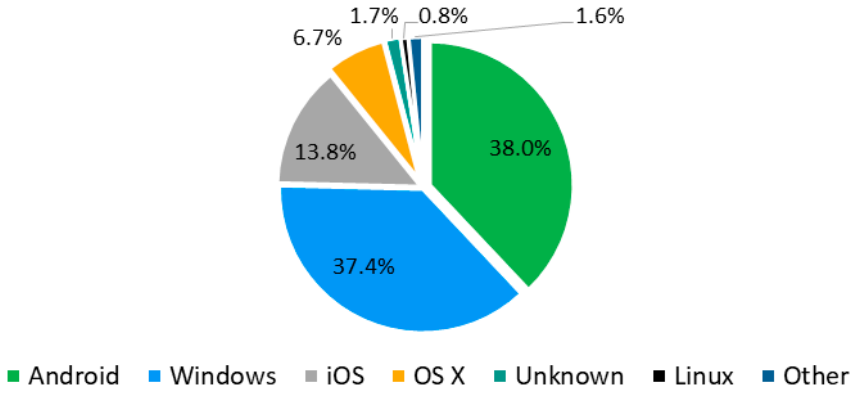

Рис. 1. Статистика долей рынка различных операционных систем.

on a chip, SoC). SoC заключается в интегрировании на одном микрочипе центрального процессора, графического ускорителя, радио-модуля и различной датчиковой аппаратуры. Данная концепция позволяет уменьшить физический размер устройства, понизить энергопотребление и повысить производительность за счет лучшей интеграции компонентов, но для взаимодействия всей системы требуется разработка специальных драйверов. Драйверы разрабатываются производителями различных чипов на кристалле и, как правило, являются проприетарными и уникальны для каждой модели. В результате, производители мобильных устройств внедряют полученные драйверы для SoC-системы в собственную сборку, что приводит к зависимости процедур обновления программного обеспечения. На рис. 2 приведен алгоритм производства мобильных устройств на базе Android. Из-за большого количества производителей чипсетов и мобильных устройств (ODM – производитель, изделия которого создаются по оригинальному проекту; OEM – производитель, детали и оборудование которого могут быть проданы другим производителям) образуется высокая степень фрагментации платформы без возможности оперативного обеспечения актуальными обновлениями мобильных устройств.

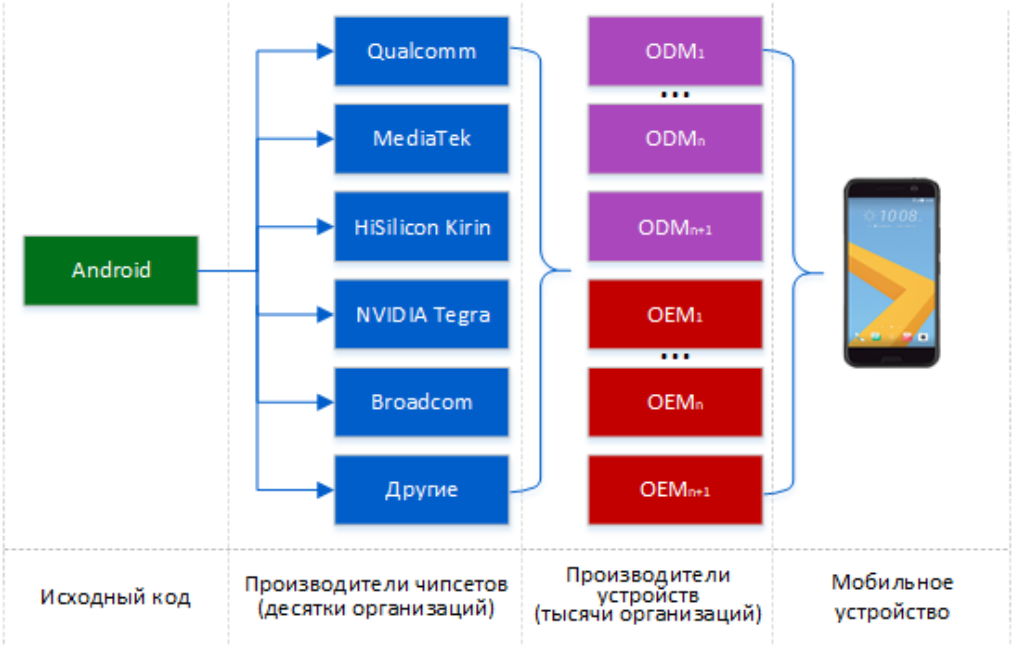

Рис. 2. Алгоритм производства устройств Android.

Компания Google для решения проблемы высокой фрагментации устройств сделала несколько важных шагов:

1. В 2014 году был представлен программно-аппаратный стандарт Android One – программа поддержки OEM-производителей, направленная на мотивирование производить устройства без модификации ОС. Целью проекта является возможность Google управлять проектированием, разработкой и поддержкой данного типа устройств, в то время как производство осуществляется производителями оригинального оборудования. В результате вопросы безопасности и обновления ОС обрабатываются Google, что позволило повысить уровень защищенности системы. На текущий момент наиболее крупные производители устройств не поддержали концепцию «чистой» ОС, что привело к низкой эффективности в решении вопроса фрагментации экосистемы.
2. В 2017 году компания Google внесла ряд изменений в архитектуру Android, добавив слой абстракции Project Treble, позволяющие отделить слой реализации аппаратного кода от кода операционной системы [2]. Данный шаг позволяет производить обновление кода операционной системы отдельно от драйверов, но получение новых версий программного обеспечения по-прежнему зависит от производителя, что качественно не улучшает ситуацию.
3. С 2018 года производители обязаны выпускать обновления ОС в течение минимум двух лет для всех Android-устройств, количество которых превышает 100 000. Если партнер не выполняет данное условие Google, возможен отказ в лицензии [3].

Политика компании Google направлена на обеспечение обновлениями устройств не старше 2-3 лет, что приводит к достаточно серьезным вопросам безопасности отрасли в целом. Соответственно, устройства, относящиеся к данному сегменту, требуется считать потенциально уязвимыми. В табл. 1 приведена краткая характеристика представленных на рынке версий Android и их статус поддержки.

Таблица 1. Характеристика версий Android

Название версии	Номер версии	Версия ядра	Год выхода
Froyo	2.2–2.2.3	2.6.32	2010
Gingerbread	2.3–2.3.7	2.6.35	2010
Honeycomb	3.0–3.2.6	2.6.36	2011
Ice Cream Sandwich	4.0–4.0.4	3.0.1	2011
Jelly Bean	4.1–4.3.1	3.0.31, 3.4.39	2012
KitKat	4.4–4.4.4	3.10	2013
Lollipop	5.0–5.1.1	3.16	2014
Marshmallow	6.0–6.0.1	3.18	2015
Nougat	7.0–7.1.2	4.4	2016
Oreo	8.0–8.1	4.10	2017
Pie	9.0	4.4.107, 4.9.84, 4.14.42	2018
– неподдерживаемая версия	– поддерживаемая версия	– актуальная версия	

На основе анализа поддерживаемых версий и статистики распределений версий Android, представленной на рис. 3, возможно вычислить вероятности нахождения устройства в уязвимом состоянии за последние 7 лет в условиях, при которых производители выполняют обновления устройств в течение двух лет [4]:

$$P_{2013} = 0.31;\ P_{2014} = 0.22;\ P_{2015} = 0.38;\ P_{2016} = 0.46;\ P_{2017} = 0.51;\ P_{2018} = 0.52;\ P_{2019} = 0.59$$

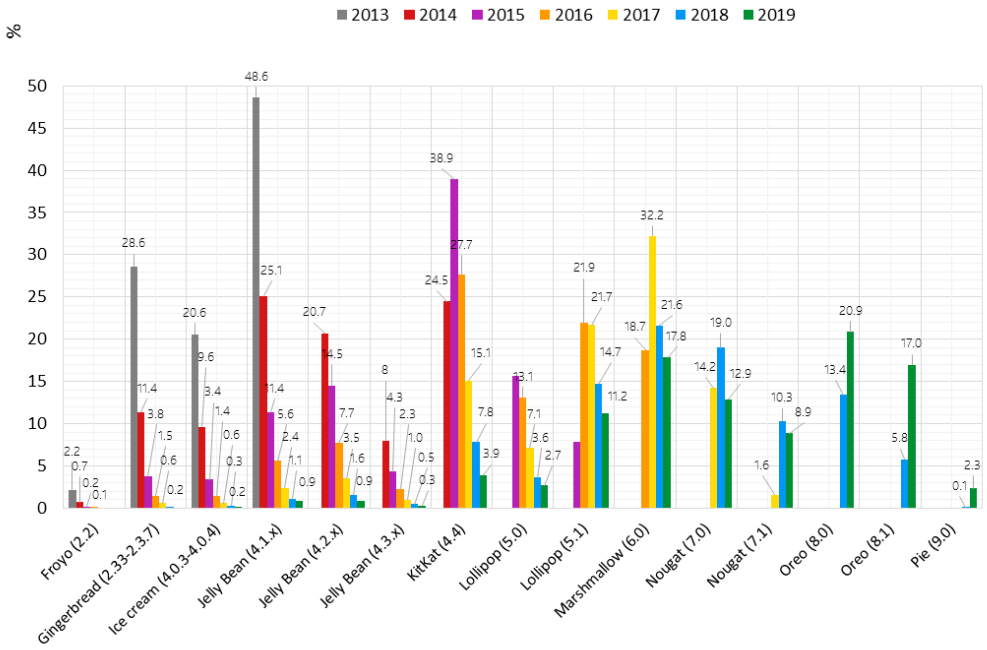

Рис. 3. Статистика распределения версий ОС Andoid.

На рис. 4 приведен график, отображающий отрицательную тенденцию распространения актуальных версий Android.

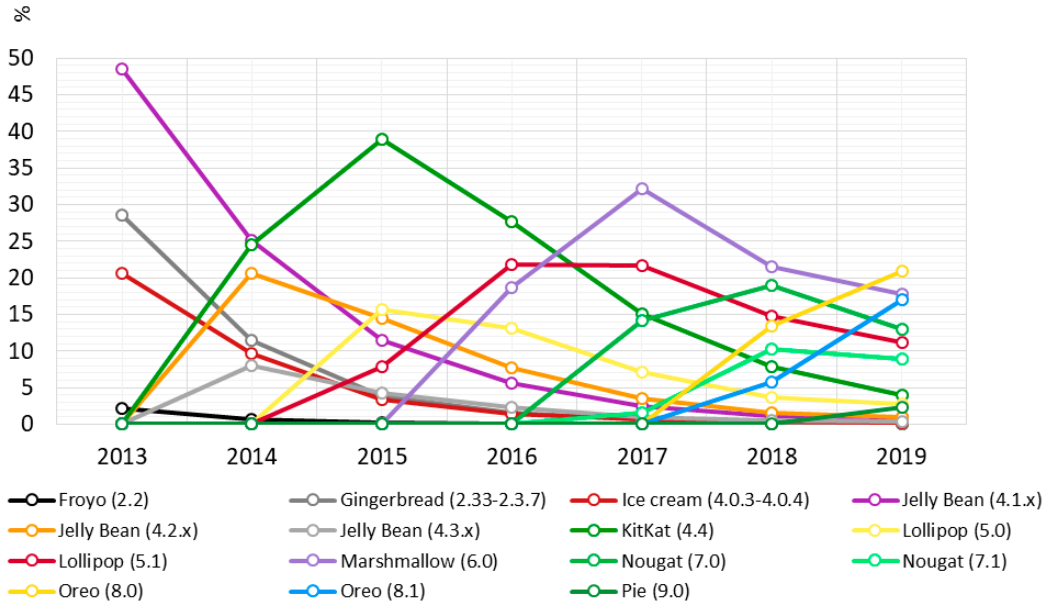

Рис. 4. График изменений количества устройств с различными версиями ОС Android.

Используя полученные данные, возможно провести прогнозирование вероятностного показателя нахождения устройства в уязвимом состоянии на ближайшие несколько лет. Для прогнозирования используется линейная функция парной регрессии (рис. 5).

Таким образом, в случае сохранения тренда развития экосистемы Android, к 2021 году вероятность компрометации устройств на данной платформе будет стремиться к 0.7,

что является серьезным сигналом в потребности создания новых методов обеспечения обновлениями безопасности мобильных устройств.

Векторы заражения

В исследовании “Android: protecting the kernel”, посвященному анализу уязвимостей Android в период с 2014 по 2016 год, отмечено резкое увеличение обнаруженных уязвимостей в ядре Linux [5]. На рис. 6 приведена статистика, характеризующая динамику роста уязвимостей ядра. В 2014 г. в ядре Android было найдено 4% уязвимостей, в 2016 г. данный показатель вырос до 36% и на 2018 год сохраняется в районе 33%.

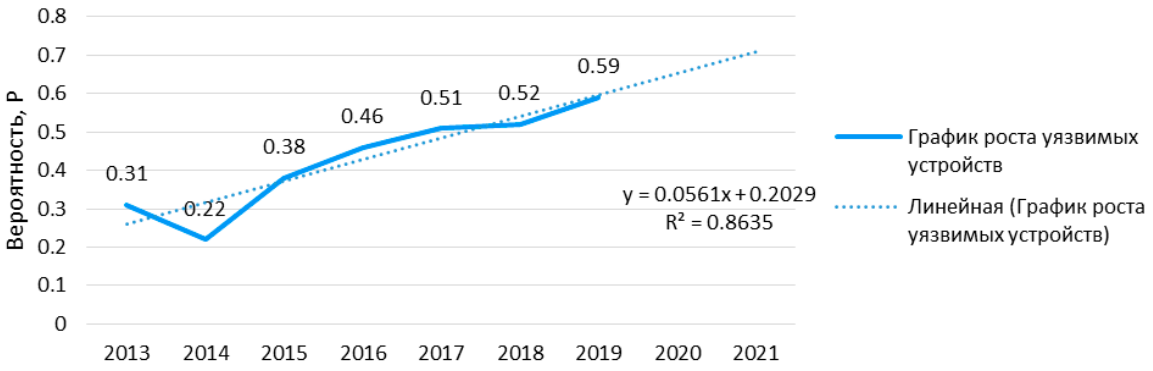

Рис. 5. График роста уязвимых устройств.

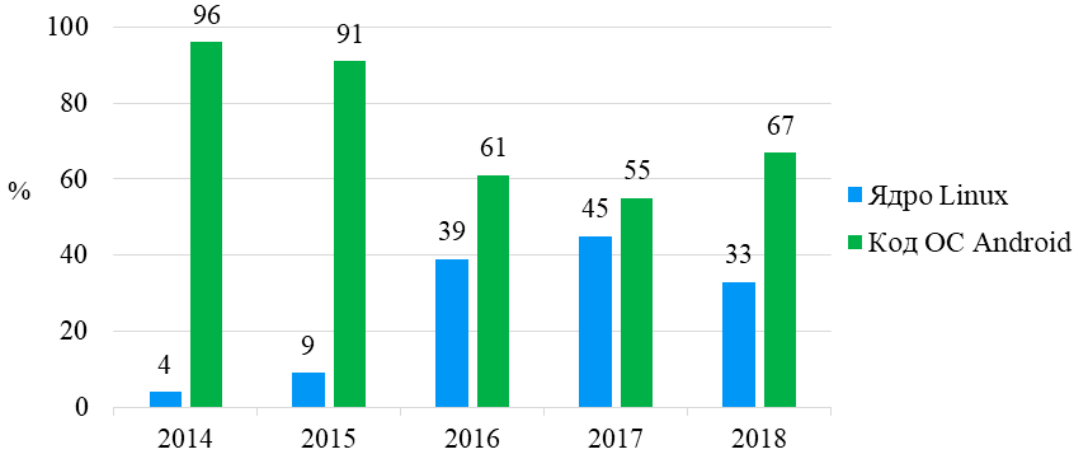

Рис. 6. Статистика уязвимостей ОС Android.

Основной причиной резкого увеличения уязвимостей в ядре является внедрение в 2013 году в версию Android 4.4 системы принудительного контроля доступа SELinux (Security Enhanced Linux), позволяющей создавать политики, определяющие виды взаимодействий, разрешенные и запрещенные для каждого процесса в рамках общего контекста безопасности, а также вводить ограничения возможностей администратора.

К ошибкам ядра относятся уязвимости не только в ядре Linux, а также в коде закрытых драйверов, поставляемых разработчиками мобильных чипсетов. На международной конференции по информационной безопасности *RSA Conference USA 2018* в докладе по исследованиям существующих уязвимостей в Android указано, что 85% уязвимостей, обнаруженных в ядре, относятся к коду закрытых драйверов [6]. На рис. 7 приведена диа-

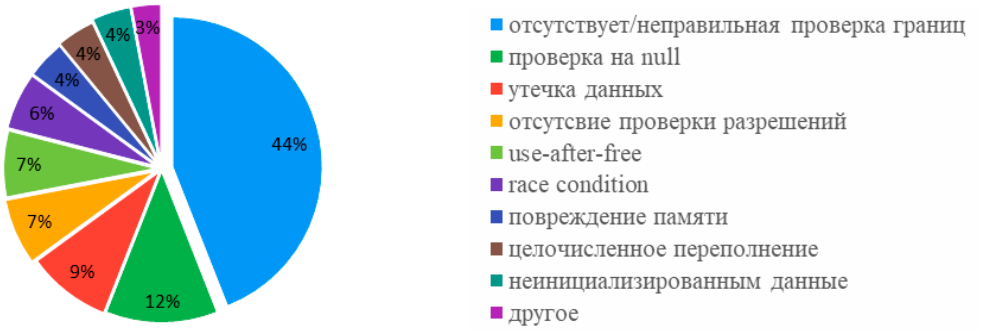

Рис. 7. Типы ошибок ядра.

грамма соотношения различных типов ошибок ядра. В результате данные типы уязвимостей способствуют проведению атак на системы независимо от версии Android.

В настоящее время каждый вид вредоносного программного обеспечения (ВПО), предназначенный для стационарных компьютеров, имеет аналог и для Android – организация бот-сети, хищение персональных данных, получение контроля над устройством или вывод его из строя и т. д. На рис. 8 приведена диаграмма, характеризующая соотношение существующих видов ВПО для Android на 3 квартал 2018 года [7]. С развитием системы разрешений и безопасности процент вирусов типа Trojan-SMS и Spyware в целом резко снизился.

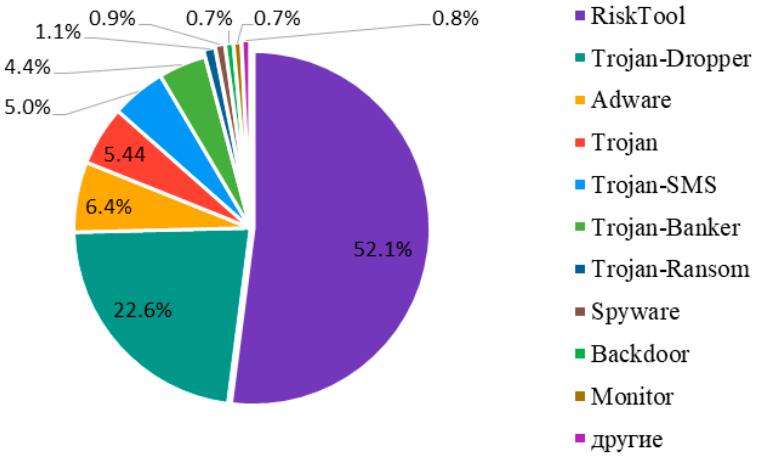

Рис. 8. Виды вредоносного программного обеспечения.

Современные технологии распространения вредоносного программного обеспечения

В настоящее время технологии распространения ВПО классифицируются следующим образом [8]:

- *Эксплуатация уязвимостей ядра Linux и его модулей.* Android является дистрибутивом Linux с собственной реализацией функций межпроцессорного взаимодействия, управления режимом «сна», защиты ядра (механизмом разделяемой памяти и т. д.) и высвобождения памяти. Таким образом, уязвимости, найденные в общих компонентах ядра, возможно применять на мобильных устройствах.

• *Эксплуатация уязвимостей аппаратных модулей.* Мобильные устройства обладают большим количеством аппаратных модулей, предназначенных для взаимодействия с другими устройствами [9]. Данный тип уязвимостей возможно эксплуатировать в зоне действия радио-модулей или при наличии непосредственного доступа к устройству.

• *Эксплуатация уязвимостей в компонентах операционных систем, программах и драйверах.* Позволяет атакующей стороне обходить средства защиты Android и SELinux.

• *Эксплуатация уязвимостей в компонентах производителей мобильных устройств.* Производители устройств выполняют модификацию Android, размещая различные приложения в директории system, т. е. запуск процессов производится в привилегированном режиме. Данные приложения могут содержать уязвимости, приводящие к утечкам данных, захвату учетных записей и установке ВПО. Также, производители сами могут модифицировать систему с не декларированным функционалом.

• *Эксплуатация уязвимостей в библиотеках.* В архитектуру Android включен ряд библиотек, таких как OpenGL, Audio Manager, Media Framework, libc и т. д. [2]. Эксплуатирование уязвимостей в данных компонентах является одним из главных векторов атаки на текущий момент. Наиболее распространенные уязвимости компонента Media Framework позволяют производить атаки типа удаленного выполнения кода (RCE) на пораженном устройстве (например, при работе с электронной почтой, просмотре сайтов в Интернете или обработке медиафайлов MMS) [10].

• *Эксплуатация уязвимостей в машинных кодах.* Android содержит инструментарий, позволяющий выполнять код C и C++ (Android NDK). Данная возможность порождает ошибки, характерные для низкоуровневых языков программирования (утечки памяти, переполнение буфера и т. д.).

• *Эксплуатация уязвимостей в пользовательских приложениях.* Приложения, установленные пользователем, могут содержать персональные данные, но хранение и доступ к данной информации не всегда обеспечивается должным образом (использование HTTP-трафика, файлы данных приложения размещены в папках с общим доступом и т. д.).

• *Использование методов социальной инженерии, применяемые для передачи и последующей установки ВПО из различных источников* (в том числе Play Market).

Способы противодействия анализу приложений, применяемые в Android

Противодействие инструментам статического анализа приложений

Для противодействия криминалистическому исследованию приложений и сигнатурному анализу используются способы маскирования, приведенные на рис. 9.

Способы различной обфускации кода направлены на приведение исходного текста или исполняемого кода программы к виду, сохраняющему ее функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

На данный момент наиболее распространенными инструментами обфускации кода приложений под Android являются DexGuard, Alatori, DashO и DexProtector. Также для первичной защиты от анализа кода применяются встроенные средства в Android Studio (Proguard или R8), но данные инструменты производят только переименование объектов. На рис. 10 приведены основные этапы преобразования исходного кода программ в исполняемый код виртуальной машины Dalvik/ART с применением данных инструментов.

В случае использования модульной архитектуры построения приложений применяются методы рефлексии, позволяющие осуществлять загрузку байт-кода в виртуальную машину во время выполнения приложения.

Рис. 9. Способы маскирования ВПО.

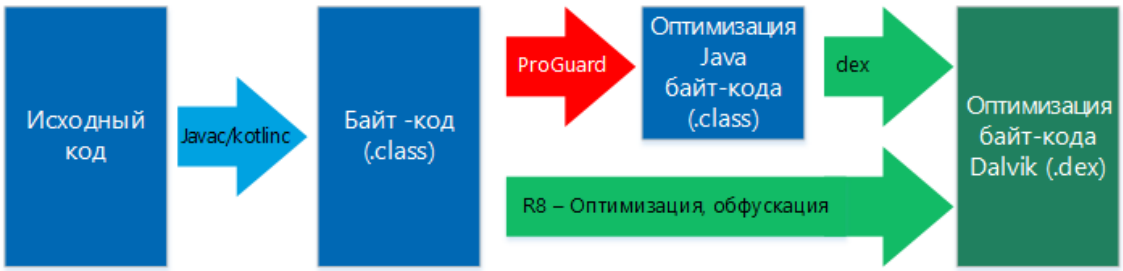

Рис. 10. Этапы создания приложений с использованием инструмента Proguard и R8.

В Android для реализации концепции динамической загрузки и исполнения кода существует следующий функционал:

1. dalvik.system.DexClassLoader – класс, позволяющий производить загрузку файлов формата «.dex», «.jar» или «.apk»;
2. Java Reflection API позволяет взаимодействовать с загруженным кодом во время выполнения программы.

Противодействие инструментам динамического анализа приложений

Способы противодействия динамическому анализу приложения, представленные на рис. 11, направлены на детектирование попыток поведенческого контроля во время выполнения программы в контролируемой «песочнице» для получения поведенческого отпечатка [11, 12]. Способ статической эвристики основан на проверке уникальных идентификаторов устройства, таких как серийный номер, принадлежность внешнего IP-адреса к сервисам проверки приложений и т. д. Как правило, при использовании гипервизора данные показатели равны значениям по умолчанию, что является показателем для де-

тектирования систем наблюдения. Способ, основанный на возникновении погрешности показаний датчиковой аппаратуры (акселерометр, гироскоп, датчик освещенности, GPS и т. д.), относят к динамической эвристике. Стоит обратить внимание, что современные эмуляторы поддерживают симуляцию датчиков с событиями обновления псевдослучайных значений, происходящими через недетерминированные интервалы времени, что затрудняет динамическую эвристику.

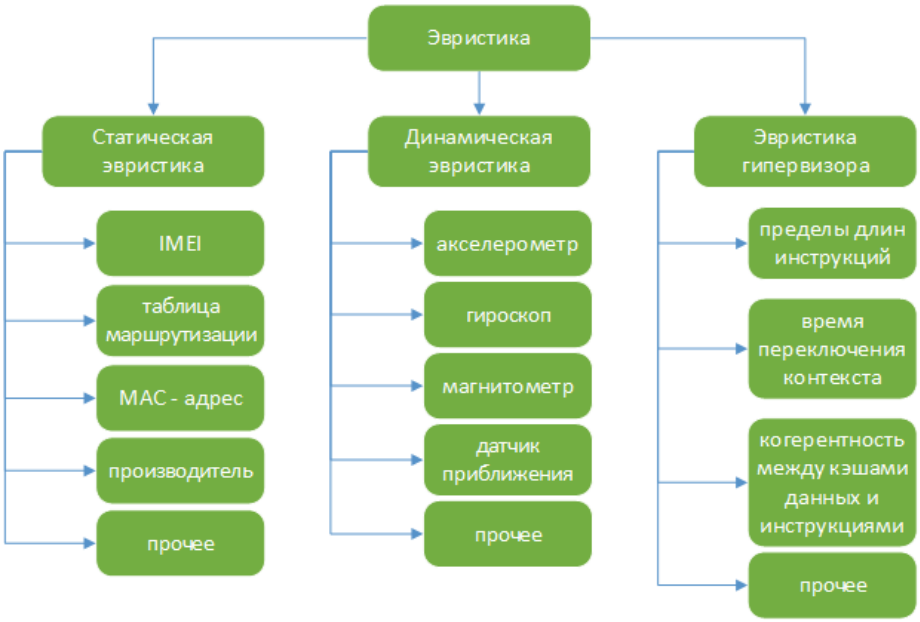

Рис. 11. Способы противодействия инструментам динамического анализа.

Эвристика гипервизора основана на различиях неполноценной эмуляции реального оборудования. Приведенные выше способы базируются на поиске различий в функционировании реальных и эмулируемых устройств. Данные отличия заключаются в наличии различий значений в регистрах MSR (моделезависимые регистры), временных интервалах переключения контекста, в пределах длин инструкций, а также специфических ошибках процессоров, относительной производительности и т. д.

Приведенные выше способы зачастую применяются для обхода системы антивирусного сканирования Google Bouncer, функционирующей в официальном репозитории Play Market, вследствие чего ВПО имеют возможность получить статус «доверенного приложения».

Современные способы и средства обеспечения безопасности в ОС Android

В Android наиболее серьезным изменениям в последних версиях была подвергнута система обеспечения безопасности на уровне платформы. Для оперативного устранения обнаруженных уязвимостей в коде Android компанией Google произведен запуск программы ежемесячного выпуска обновлений безопасности. Начиная с Android 8.0, добавлен компонент на уровне платформы “Play Protect” – облачная система безопасности, обеспечивающая сканирование приложений в официальном репозитории и на устройствах пользователей в режиме реального времени.

В табл. 2 перечислены современные средства обеспечения информационной безопасности на уровне устройства и их назначение [13, 14].

Таблица 2. Механизмы защиты Android

№	Функция безопасности	Описание типа защиты
1.	Шифрование	Защита данных от несанкционированного доступа с использованием средств криптографической защиты информации.
2.	Аппаратные средства защиты	Обеспечение надежного хранения ключей шифрования и защищенной процедуры аутентификации.
3.	Защита ядра	PAN (Privileged Access Never) – запрет обращения к памяти процессов напрямую и принуждение использования функций копирования памяти (Android 8.0); CFI (Control Flow Integrity) – создание графа вызовов функций, встраивание кода сверки с данным графом перед каждым вызовом функций. Данный механизм направлен на противодействие модификации указателей на функцию и адресов возврата (Android 9.0); IOS (Integer Overflow Sanitization) – защита данных от целочисленное переполнение. Проверка выполнения арифметических операций (Android 7.0); KASLR (Kernel Address Space Layout Randomization) – присвоение случайного адреса памяти участку расположения кода, области анонимной памяти, данных ядра в памяти при каждой загрузке (Android 8.0); PIROM (Post-Init Read-Only Memory) – создание доступных для чтения и записи областей памяти, используемых только во время инициализации и переводимых в режим «только для чтения» после инициализации (Android 8.0).
4.	Изоляция процессов	Выполнение каждого приложения в отдельном адресном пространстве с уникальными значениями user/group ID. Осуществление обмена данными между процессами осуществляется через сервисы ОС.
	SELinux	Принудительный контроль доступа, обеспечивающий минимально необходимый набор привилегий для каждого процесса.
5.	Верификация загрузки ОС	Обеспечение исправного состоянии операционной системы при загрузке. Механизмы, предотвращающие запуск самоподписанного кода и проверки целостности и цифровой подписи ядра.

Заключение

Проблемы в вопросах обеспечения безопасности платформы Android присутствуют на всех уровнях платформы. В современных версиях Android механизмы защиты серьезно повысили степень защищенности устройств, но данные решения не направлены на обеспечение безопасности экосистемы в целом и не позволяют гарантировать абсолютную надежность и безопасность данных в мобильных устройствах. Несмотря на значительное повышение качества механизмов защиты, количество уязвимостей продолжает увеличиваться. Согласно данным с ресурса cvedetails.com, на текущий момент в Android насчитывается 2146 уязвимостей, причем в 2017 г. обнаружено 842, а в 2018 – 611 уязвимостей [15].

Высокая степень фрагментации платформы и прогнозируемое увеличение числа уязвимых устройств требует применения новых методов обеспечения безопасности мобильных операционных систем и систем анализа на наличие вредоносного кода. Данная область представляет важную и актуальную задачу с необходимостью детальной проработки вопросов идентификации вредоносного кода.

Благодарности

Работа выполнена при поддержке Министерства образования и науки Российской Федерации (государственный контракт № 03.G25.31.0280 от 30 мая 2017 г.).

Литература / References:

1. Operating System Market Share Worldwide. URL: http://gs.statcounter.com/os-market-share (accessed March 02, 2019).

2. Android Architecture. URL: https://source.android.com/devices/architecture (accessed January 02, 2019).

3. Android Enterprise Recommended. URL: https://www.android.com/intl/ru_ru/enterprise/recommended/ (accessed January 02, 2019).

4. Android version market share distribution among smartphone owners as of September 2018. URL: https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/ (accessed February 23, 2019).

5. Stoep J.V. Android: protecting the kernel. – 2016, 35 p. URL: https://events.linuxfoundation.org/sites/events/files/slides/Android%20protecting%20the%20kernel.pdf (accessed March 03, 2019).

6. Vigna G. How automated vulnerability analysis discovered hundreds of Android 0-days. URL: https://www.rsaconference.com/writable/presentations/file_upload/mbs-r14-how-automated-vulnerability-analysis-discovered-hundreds-of-android-0-days.pdf (accessed January 02, 2019).

7. Chebyshev V. Mobile malware evolution 2018. URL: https://securelist.com/mobile-malware-evolution-2018/89689/ (accessed March 15, 2019).

8. Романеев М. Проблемы безопасности Android-приложений: классификация и анализ [Romaneev M. Android Application Security Issues: Classification and Analysis]. URL: http://www.pvsm.ru/android/260205#17 (accessed February 27, 2019).

9. Kaspersky_Lab Security Week 45: Кое-что об уязвимостях в Bluetooth [Something about Bluetooth vulnerabilities]. URL: http://www.pvsm.ru/android/298251 (accessed February 10, 2019).

10. CVE-2017-0466 Detail. URL: https://nvd.nist.gov/vuln/detail/CVE-2017-0466 (accessed April 22, 2019).

11. Petsas T., Voyatzis G., Athanasopoulos E., Ioannidis S., Polychronakis M. Rage against the virtual machine: Hindering dynamic analysis of Android malware [presentation]. URL: http://www.syssec-project.eu/m/documents/eurosec14/RATVM.pdf (accessed November 18, 2018).

12. Petsas T., Voyatzis G., Athanasopoulos E., Ioannidis S., Polychronakis M. Rage against the virtual machine: Hindering dynamic analysis of Android malware URL: http://www.syssec-project.eu/m/page-media/3/petsas_rage_against_the_virtual_machine.pdf (accessed March 02, 2019).

13. Android Security & Privacy 2018 Year in Review. URL: https://source.android.com/security/reports/Google_Android_Security_2018_Report_Final.pdf (accessed March 02, 2019).

14. Security Blog. URL: https://security.googleblog.com/search/label/android%20security (accessed April 02, 2019).

15. Vulnerability Trends Over Time. URL: https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224 (accessed April 25, 2019).

Об авторах:

Изергин Дмитрий Андреевич, ассистент кафедры КБ-4 «Интеллектуальные системы информационной безопасности» Института комплексной безопасности и специального приборостроения ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). https://orcid.org/0000-0002-3174-4550

Еремеев Михаил Алексеевич, доктор технических наук, профессор, профессор кафедры КБ-2 «Прикладные информационные технологии» Института комплексной безопасности и специального приборостроения ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). https://orcid.org/0000-0002-5511-4000

Магомедов Шамиль Гасангусейнович, кандидат технических наук, доцент кафедры КБ-4 «Интеллектуальные системы информационной безопасности» Института комплексной безопасности и специального приборостроения ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). https://orcid.org/0000-0001-8560-1937. ResearcherID M-5782-2016

Смирнов Станислав Игоревич, старший преподаватель кафедры КБ-4 «Интеллектуальные системы информационной безопасности» Института комплексной безопасности и специального приборостроения ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). https://orcid.org/0000-0003-4387-0850

About the authors:

Dmitriy A. Izergin, Assistant of Professor of the Chair CS-4 “Intelligent Information Security Systems”, Institute of Integrated Safety and Special Instrumentation, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). https://orcid.org/0000-0002-3174-4550

Mikhail A. Eremeev, Dr. of Sci. (Engineering), Professor, Professor of the Chair CS-2 “Applied Information Technologies”, Institute of Integrated Safety and Special Instrumentation, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). https://orcid.org/0000-0002-5511-4000

Shamil G. Magomedov, Cand. of Sci. (Engineering), Associate Professor of the Chair CS-4 “Intelligent Information Security Systems”, Institute of Integrated Security and Special Instrumentation, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). https://orcid.org/0000-0001-8560-1937. ResearcherID M-5782-2016

Stanislav I. Smirnov, Senior Lecturer of the Chair CS-4 “Intelligent Information Security Systems”, Institute of Integrated Security and Special Instrumentation, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia). https://orcid.org/0000-0003-4387-0850