Научная статья
УДК 004.056
DOI: 10.26583/bit.2025.3.12

ВОЗМОЖНОСТИ ОБНАРУЖЕНИЯ ВРЕДОНОСНОГО ПО ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ ПОД УПРАВЛЕНИЕМ ОС ANDROID С ПОМОЩЬЮ МОНИТОРИНГА ПОТРЕБЛЕНИЯ АППАРАТНЫХ РЕСУРСОВ

Андрей М. Бонч-Бруевич[1], Алена А. Борисовская[2]
Московский государственный лингвистический университет, ул. Остоженка дом 38 стр.1, Москва, 119034, Россия
[1]123andryb@mail.ru, https://orcid.org/0000-0002-4453-2979
[2]borisovsk0101@gmail.com, https://orcid.org/0009-0006-0746-0781

Аннотация. Целью статьи является исследование возможностей обнаружения вредоносного программного обеспечения (ПО) на устройствах под управлением операционной системы Android на основе анализа статистики использования аппаратных ресурсов. Современные вредоносные приложения демонстрируют разнообразные паттерны потребления ресурсов, что позволяет выявлять их аномальное поведение. В рамках работы рассмотрено, как вредоносное ПО влияет на работу устройства, как проявляет себя в фоновом режиме и по каким признакам может быть обнаружено. В отличие от традиционных сигнатурных методов, мониторинг параметров использования процессора, оперативной памяти, сетевой активности и энергопотребления позволяет выявлять сложные вредоносные мобильные приложения. Приведены примеры вредоносных приложений, таких как трояны-майнеры, рекламные трояны и шпионские программы, а также проведен анализ методов их выявления на основе мониторинга ключевых индикаторов работы устройства. Рассмотрены такие подходы к их обнаружению, как динамический анализ в песочнице, использование встроенных утилит операционной системы, эмуляция среды выполнения для выявления скрытых угроз. Рассмотрены ограничения и недостатки подхода на основе мониторинга потребления аппаратных ресурсов, включая высокую нагрузку на систему и возможные ложные срабатывания. Для повышения эффективности обнаружения предложены модификации существующих методов – введение дополнительных метрик анализа, динамическая корректировка интервала сбора данных в зависимости от нагрузки на устройство, применение машинного обучения для классификации аномалий. Данные усовершенствования могут снизить количество ложных срабатываний и повысить точность детектирования, что показывает перспективность мониторинга аппаратных ресурсов в качестве дополнения к традиционным антивирусным подходам.

Ключевые слова: *вредоносное ПО, мобильные устройства, Android, мониторинг аппаратных ресурсов, энергопотребление.*

Для цитирования. *Бонч-Бруевич, Андрей М.; Борисовская, Алена А. Возможности обнаружения вредоносного по для мобильных устройств под управлением ОС Android с помощью мониторинга потребления аппаратных ресурсов. Безопасность информационных технологий, [S.l.], т. 32, № 3, с. 146–154, 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1816. DOI: http://dx.doi.org/10.26583/bit.2025.3.12.*

Scientific article

MALWARE DETECTION CAPABILITIES FOR MOBILE DEVICES RUNNING ANDROID OS BY MONITORING HARDWARE RESOURCE CONSUMPTION

Andrew M. Bonch-Bruevich[1], Alyona A. Borisovskaya[2]
Moscow State Linguistic University, Ostozhenka str., 38, building 1, Moscow, 119034, Russia
[1]123andryb@mail.ru, https://orcid.org/0000-0002-4453-2979

[2]*borisovsk0101@gmail.com, https://orcid.org/0009-0006-0746-0781*

Abstract. The aim of the article is to explore the potential for detecting malicious software (malware) on devices running the Android operating system by analyzing hardware resource usage statistics. Modern malicious applications exhibit diverse resource consumption patterns, enabling the detection of their anomalous behavior. This study examines how malware affects device performance, operates in the background, and the indicators that can reveal its presence. Unlike traditional signature-based methods, monitoring parameters such as CPU usage, RAM utilization, network activity, and power consumption allows for the detection of sophisticated malicious mobile applications. Examples of malware, such as trojan miners, adware trojans, and spyware, are provided, along with an analysis of detection methods based on monitoring key device performance indicators. Approaches to their detection, including dynamic analysis in sandbox environments, the use of built-in OS utilities, and runtime environment emulation to uncover hidden threats, are discussed. The limitations and drawbacks of hardware resource monitoring - such as high system overhead and potential false positives - are also examined. To improve detection effectiveness, modifications to existing methods are proposed, such as introducing additional analysis metrics, dynamically adjusting data collection intervals based on device load, and applying machine learning for anomaly classification. These enhancements could reduce false positives and increase detection accuracy, making hardware resource monitoring a more promising complement to traditional antivirus approaches.

Keywords: *malware, mobile devices, Android, hardware monitoring, power consumption.*

For citation: *Bonch-Bruevich, Andrew M.; Borisovskaya, Alyona A. Malware detection capabilities for mobile devices running Android OS by monitoring hardware resource consumption. IT Security (Russia), [S.l.], v. 32, no. 3, p. 146–154, 2025. ISSN 2074-7136 URL: https://bit.spels.ru/index.php/bit/article/view/1816. DOI: http://dx.doi.org/10.26583/bit.2025.3.12.*

Введение

Android, как операционная система с открытым исходным кодом, занимает доминирующее положение на мировом рынке мобильных устройств, что обусловлено её гибкостью и возможностью установки приложений из различных источников, включая неофициальные магазины приложений. Однако это создаёт серьёзные риски для информационной безопасности пользователей. По данным последних исследований[1], количество вредоносных приложений для Android растёт экспоненциально, при этом традиционные методы защиты, основанные на сигнатурном анализе, становятся всё менее эффективными. Это связано как с постоянным появлением новых модификаций вредоносного кода, так и с использованием злоумышленниками техник обфускации, позволяющих обходить статические проверки.

Современные вредоносные приложения демонстрируют широкий спектр функциональности – от скрытого майнинга криптовалют до кражи конфиденциальных данных пользователей. Общим для некоторых таких угроз является их специфическое воздействие на аппаратные ресурсы устройства: аномальное потребление процессорного времени, повышенное использование оперативной памяти, нехарактерная активность сетевого интерфейса и ускоренный разряд аккумулятора [1].

Эти особенности позволяют выдвинуть гипотезу о том, что мониторинг параметров использования аппаратных ресурсов может служить эффективным методом обнаружения вредоносной активности, дополняющим традиционные подходы.

[1]«Dr.Web»: обзор вирусной активности для мобильных устройств за 2024 г. URL: https://news.drweb.ru/show/review/?lng=ru&i=14970 (дата обращения: 10.04.2025).

Целью данной работы является комплексный анализ возможностей обнаружения вредоносного ПО на устройствах под управлением ОС Android посредством мониторинга потребления аппаратных ресурсов. В рамках исследования проведен анализ воздействия современных вредоносных приложений на потребление устройством системных ресурсов, а также разработаны критерии аномального поведения приложений на основе анализа потребления CPU, RAM, сетевого трафика и энергопотребления. Проведена оценка эффективности различных методов и инструментов мониторинга потребления аппаратных ресурсов, выявлены ограничения и недостатки этого подхода и предложены пути модификации данного метода для эффективного автоматического обнаружения вредоносной активности.

Актуальность исследования обусловлена необходимостью разработки новых, более эффективных методов защиты Android-устройств, способных противостоять эволюционирующим угрозам. Практическая значимость работы заключается в возможности создания на её основе новых систем мониторинга безопасности, не требующих постоянного обновления сигнатурных баз и способных обнаруживать ранее неизвестные вредоносные приложения.

1. Влияние современных вредоносных приложений на поведение мобильного устройства

Современные вредоносные приложения для Android демонстрируют различные паттерны потребления аппаратных ресурсов, на основе которых возможно выявить их аномальную активность.

Например, троянцы-майнеры (CoinHive) [2] потребляют до 80% мощности CPU в фоновом режиме, вызывая перегрев устройства и сокращение времени автономной работы на 30–50%. Шпионские приложения, такие как Triout, скрыто активируют микрофон и камеру, что может привести к повышенному энергопотреблению и использованию оперативной памяти (до 500 МБ дополнительной нагрузки). Ботнеты (например, AndroRAT [3]) генерируют аномальный сетевой трафик – до 1 ГБ/сутки на устройство – для передачи данных на CnC-серверы, что снижает скорость интернет-соединения.

Статистические данные подтверждают, что рекламные трояны (например, HiddenAds [4] увеличивают нагрузку на GPU из-за постоянной демонстрации баннеров, что в 40% случаев приводит к зависаниям интерфейса. Банковские вредоносы (SpyNote, Anatsa [5]) активно используют фоновые процессы для перехвата SMS, что сопровождается скачкообразным ростом потребления CPU (15–25%) при получении сообщений.

Экспериментальные исследования (проект AndroDFA [1]) показывают, что динамический анализ ресурсных метрик (CPU, RAM, трафик) позволяет классифицировать 82% известных семейств вредоносного ПО с точностью до 78% на физических устройствах.

Однако, при очевидной целесообразности мониторинга ресурсных аномалий многие гибридные угрозы (например, Joker [6]) могут плохо поддаваться выявлению из-за использования различных механизмов маскировки, в частности, под системные процессы.

Рассмотрим далее основные способы, которыми вредоносные приложения могут маскироваться на зараженном мобильном устройстве.

2. Механизмы маскировки вредоносных приложений

Вредоносные приложения, в особенности, майнеры, применяют различные способы маскировки на мобильном устройстве, чтобы их нельзя было обнаружить методами мониторинга потребления ресурсов.

Один из таких способов – снижение активности при обнаружении факта мониторинга, а также адаптивное увеличение и снижение нагрузки. Например, HiddenMiner и Coinhive запускают майнинг только при зарядке устройства или в ночное время [7]. Также они ограничивают нагрузку на CPU до 10–15%, чтобы не вызывать перегрев, и отключаются при падении заряда батареи ниже 20%.

Вредоносные приложения также могут внедряться в легитимные системные процессы (com.android.systemui, com.google.process.gapps) и использовать поддельные имена пакетов, похожие на системные (например, googleupdateservice.apk). Такое поведение демонстрируют троянские приложения Triada и xHelper [8]. То есть мониторинг потребления аппаратных ресурсов не обнаружит подозрительных процессов и сервисов, что снизит эффективность данного метода.

Вредоносный код может не оставлять следов в APK файле, а загружаться приложением через WebView, например, в фоновом браузере. Так работает майнер CoinIMP, основные действия которого происходят через JavaScript-скрипты, следовательно, выносятся из процесса основного приложения и трудно выявляются при мониторинге.

Наиболее характерный способ маскировки, который используют вредоносные приложения для избежания обнаружения при мониторинге – имитация легитимной деятельности. Многие майнеры имитируют функционал видеоплееров или VPN-приложений, маскируя трафик под легитимный (Vilny.net) [9]. А также используют легальные SDK (например, рекламные модули) для скрытия вредоносных вызовов и успешно проходят проверки безопасности GooglePlay, проникая в официальные магазины приложений.

Использование TrustZone и root-привилегий демонстрируют некоторые трояны для скрытия факта присутствия в системе от средств мониторинга. TrustZone [10] – это изолированная среда выполнения (TEE, Trusted Execution Environment), предназначенная для защиты критических операций. Однако вредоносное ПО часто использует её для отключения логирования через модификацию syslogd и скрывается таким образом от инструментов мониторинга потребления системных ресурсов.

Подводя итог вышеперечисленному, современные вредоносные приложения, целью которых является выполнение ресурсоёмких операций (майнинг, реклама, шпионское ПО), потенциально детектируемы средствами мониторинга потребления системных ресурсов, но умеют скрываться от данных инструментов, следовательно, методы мониторинга должны быть усовершенствованы и адаптированы к актуальным угрозам.

3. Методы мониторинга потребления системных ресурсов

Рассмотрим и сравним существующие методы мониторинга потребления системных ресурсов мобильными приложениями, выявим их основные недостатки и потенциальные пути модификации.

Классические утилиты операционной системы Android, такие как adb dumpsys[2], top, procstats, позволяют осуществлять сбор данных о процессах и потреблении системных ресурсов. Эти инструменты предоставляют детализацию данных о приложениях по их UID, чего во многих случаях достаточно для детектирования вредоносного приложения. Однако, эти инструменты требуют модифицированного подхода для вредоносных приложений с адаптивной нагрузкой и скрывающихся под легитимные и системные процессы.

[2]Android System Resource Monitoring. URL: https://docs.digital.ai/continuous-testing/docs/te/test-execution-home/mobile-android-and-ios/appium/seetest-appium-extension/monitoring-commands/android-system-resources-monitoring (дата обращения: 10.04.2025).

Многие инструменты динамического анализа, работающие в фоне, основаны на использовании системного Android API: Battery Historian, Batterystats [11]. Но они обладают ограничениями: не предоставляют детализацию по отдельным процессам (приложениям) без root-прав. То есть инструмент мониторинга на его основе должен обладать в системе повышенными привилегиями. Аналогично, системные инструменты не учитывают адаптивную нагрузку и нуждаются в добавлении временных окон анализа (например, сравнение ночной и дневной активности).

Распространённым инструментом мониторинга поведения приложений является динамический анализ в песочнице. При этом запуск приложений происходит в изолированной среде: эмуляторе или облачной песочнице. В ходе мониторинга фиксируются системные вызовы и нагрузки на ресурсы. Реализовать такой мониторинг можно в Google Play Protect Sandbox с помощью инструмента CuckooDroid [12]. Такой метод может успешно выявить скрытый майнинг через CPU/GPU.

Это более гибкий подход, но у него есть следующие недостатки: факт запуска приложения в эмуляторе детектируется некоторыми вредоносными приложениями. Также при активном анализе происходит высокая нагрузка на серверы.

Использование TrustZone (TEE) для аппаратного мониторинга, например, считывания показателей датчиков контроля температуры CPU/GPU и энергопотребления. ARM TrustZone Monitor [13] и QSEE-логи могут фиксировать аномалии энергопотребления и факты несанкционированного доступа к критическим ресурсам мобильного устройства. Недостатком этого подхода является требование наличия root-доступа, а также неприменимость для бюджетных устройств без TEE (табл. 1).

Таблица 1. Сравнительный анализ инструментов мониторинга потребления системных ресурсов

Инструмент	Применимость	Недостатки
dumpsys	Любой Android без root	Ложные срабатывания на фоновые процессы
Battery Historian	Требует ADB и Python	Нет детализации по PID без root
Google Play Protect Sandbox + CuckooDroid	Требует серверной инфраструктуры	Эмуляция обнаруживается продвинутыми приложениями
ARM TrustZone Monitor + QSEE-логи	Требует root, TEE	Доступно только на устройствах с TrustZone

Подводя итог, для базового мониторинга возможно использовать подходы на основе инструментов dumpsys и Battery Historian, для более глубокого анализа подозрительных приложений возможно использовать песочницу Google Play Protect Sandbox и CuckooDroid, а для корпоративной защиты лучшие показатели обнаружения вредоносного ПО можно обеспечить с помощью мониторинга внутри TrustZone (TEE). Однако главные недостатки существующих инструментов динамического мониторинга аппаратных ресурсов – ложные срабатывания, невозможность обнаружить адаптирующиеся угрозы и сложность практического применения.

4. Общие ограничения подхода на основе ресурсного мониторинга

Несмотря на перспективность методов обнаружения вредоносного ПО на основе анализа потребления ресурсов (CPU, RAM, энергопотребление, сетевой трафик), этот подход имеет некоторые ограничения, общие для всех современных инструментов мониторинга.

Первый фактор – высокий уровень ложных срабатываний: легитимные приложения, такие как видеоплееры или игры, могут демонстрировать схожие с вредоносными программами паттерны нагрузки на CPU и батарею, что снижает точность детектирования. Например, исследование AndroDFA показало, что даже при использовании машинного обучения точность классификации достаточно низкая для новых угроз [1].

Кроме того, адаптивность вредоносного ПО: современные вредоносные приложения, такие как Joker или Triada, снижают нагрузку на мобильное устройство в моменты мониторинга или используют периодическую активность, чтобы избежать обнаружения. Например, троян DroidJack [14] активируется только при определённых условиях (при запуске конкретного приложения), что делает его незаметным для систем, полагающихся на ресурсный мониторинг.

Ещё одно важное ограничение – зависимость от окружения: динамический анализ в эмуляторах (DroidBox, GP Protect Sandbox) часто не отражает поведение вредоносных приложений на реальных устройствах, так как многие из них детектируют виртуальную среду и меняют своё поведение. Эксперименты с набором данных KronoDroid подтвердили, что разница в метриках ресурсов между эмулятором и физическим устройством может достигать 30%, что критично для точности классификации [15]. Кроме того, энергоэффективные алгоритмы Android (Doze Mode) искажают данные мониторинга, маскируя фоновую активность вредоносных процессов.

Также играет роль сложность анализа многофункциональных угроз, так как вредоносные приложения (Anubis) могут комбинировать различные тактики шифрования трафика и минимальной нагрузки на CPU, что требует комплексного подхода, выходящего за рамки ресурсного мониторинга. Исследование DroidScribe [16] показало, что даже продвинутые методы на основе машинного обучения достигают точности лишь 78% для гибридных угроз.

Таким образом, хотя анализ ресурсных паттернов остаётся эффективным, его применимость ограничена необходимостью интеграции с другими методами из-за механизмов сокрытия вредоносным ПО своего присутствия в системе. Следовательно, актуальна задача дополнения методов ресурсного мониторинга функциями, позволяющими адаптироваться к совершенствующимся угрозам.

5. Модификация и дополнение метода на основе мониторинга потребления аппаратных ресурсов

Как обсуждалось ранее, подход к обнаружению вредоносных приложений под Android на основе мониторинга паттернов потребления аппаратных ресурсов требует решения ряда ключевых ограничений, каждое из которых обусловлено как техническими ограничениями платформы, так и совершенствующимися угрозами.

Поскольку традиционные инструменты мониторинга часто требуют root-прав, это ограничивает их применение на пользовательских устройствах. Для обхода этого ограничения целесообразно использовать API и прокси-методы доступа к системным ресурсам без необходимости получения root-доступа, например, через ADB run-as для собственных приложений или создание прокси-классов для ключевых системных сервисов[3].

[3]Access the Data/Data Folder in Android Devices Without Rooting. URL: https://www.repeato.app/how-to-access-the-data-data-folder-in-android-devices-without-rooting/ (дата обращения: 10.04.2025).

Как было показано ранее, вредоносные приложения могут адаптивно изменять нагрузку на систему, чтобы избежать обнаружения. Решением является динамическая корректировка интервалов сбора данных и применение адаптивных стратегий мониторинга, позволяющих фиксировать кратковременные аномалии и учитывать изменчивость поведения приложений [17].

Вследствие маскировки вредоносных программ под легитимные или системные процессы требуется применение многоуровневой фильтрации, машинного обучения и тестирования в различных сценариях. Например, использование ансамблей моделей и алгоритмов отбора признаков позволяет повысить точность классификации и объяснимость решений, что подтверждено экспериментами с точностью до 98,8% на реальных выборках [18].

Поскольку многие вредоносные приложения способны обнаруживать запуск в песочнице или эмуляторе и изменять свое поведение, необходимо маскировать артефакты виртуализации, имитировать пользовательскую активность и наполнять окружение реалистичными пользовательскими данными, что реализовано, например, в платформе CamoDroid [19].

Кроме того, высокий уровень ложных срабатываний систем мониторинга на легитимные приложения с интенсивным энергопотреблением можно снизить. Для этого можно использовать корреляцию показателей энергопотребления с дополнительными метриками: сетевой активностью, обращениями к API, профилями нагрузки на CPU и память. Такой подход позволяет отличать аномальное поведение от типичного для ресурсоемких легитимных приложений [20].

В совокупности, применение перечисленных модификаций может обеспечить большую устойчивость, масштабируемость и точность системе ресурсного мониторинга для выявления вредоносных приложений на Android.

При этом остаются нерешёнными задачи оптимизации энергопотребления самой системы мониторинга, регулярного обновления моделей машинного обучения для поддержания актуальности и устойчивости к новым угрозам, а также ограничения, связанные с политиками доступа к системным данным в новых версиях Android. Это требует дальнейших исследований и совершенствования архитектуры систем мониторинга [21].

Заключение

В данной работе проведен анализ современных вредоносных приложений для Android и их паттернов потребления системных ресурсов мобильного устройства. Рассмотрены способы маскировки в системе современных вредоносных приложений, что делает их незаметными для обычного ресурсного мониторинга. Проведена оценка ограничений и недостатков существующих инструментов и в целом подхода к обнаружению вредоносного ПО на мобильном устройстве на основе мониторинга потребления системных ресурсов. Проанализированы возможные подходы к увеличению точности обнаружения вредоносного ПО на основе существующих инструментов, а также пути их модификации для создания эффективной комплексной системы мониторинга потребления системных ресурсов для детектирования вредоносных приложений.

СПИСОК ЛИТЕРАТУРЫ:

1. Massarelli, L., Aniello, L., Ciccotelli, C., Querzoni, L., Ucci, D., Baldoni, R. AndroDFA: Android Malware Classification Based on Resource Consumption. Information. 2020. 11(6), 326. DOI:10.48550/arXiv.1709.00875.
2. Dashevskyi, S., Zhauniarovich, Y., Gadyatskaya, O., Pilgun, A., Ouhssain, H. Dissecting Android Cryptocurrency Miners. Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy. 2020, p. 191–202. DOI: 10.48550/arXiv.1905.02602.

3. Zhi, L.H., Selvarajah, V. Android Phone Hacking and Impact via Androrat. AIP Conference Proceedings. 2024, v. 2802, no. 1. DOI:10.1063/5.0181916.
4. Suresh, S., Di Troia, F., Potika, K. et al. An analysis of Android adware. J Comput Virol Hack Tech 15, p. 147–160 (2019). DOI: https://doi.org/10.1007/s11416-018-0328-8.
5. PranavRaj, R., Syam, S.R. Identification Of Malware Families Using Energy Consumption. 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). 2024, p. 1–6. DOI: 10.1109/ICCCNT61001.2024.10724670.
6. Shi, C., Cheng, C.C.C., Guan, Y. Forensic Analysis on Joker Family Android Malware. 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). 2021, p. 403–406. DOI:10.1109/WiMob52687.2021.9606310.
7. Rahman, M., Rahman, M., Carbunar, B., Chau, D.H. Search Rank Fraud and Malware Detection in Google Play. IEEE Transactions on Knowledge and Data Engineering. 2017, 29(6), p. 1329–1342. DOI: 10.1109/TKDE.2017.2667658.
8. Kumar, K.A., Raman, A., Gupta, C., Pillai, R.R. The Recent Trends in Malware Evolution, Detection and Analysis for Android Devices. Journal of Engineering Science & Technology Review. 2020, 13(4). DOI: 10.25103/jestr.134.25.
9. Arshad, S., Shah, M.A., Khan, A., Ahmed, M. Android Malware Detection & Protection: A Survey. International Journal of Advanced Computer Science and Applications. 2016. 7(2). DOI: 10.14569/IJACSA.2016.070262.
10. Yalew, S.D. Mobile Device Security with ARM TrustZone. KTH Royal Institute of Technology, 2018. URL: https://www.semanticscholar.org/paper/Mobile-Device-Security-with-ARM-TrustZone-Yalew/d39f714e82724fb1d2352daf8e3814ee94fd8851 (дата обращения: 10.04.2025).
11. Chung, J., Jung, H., Koo, J., Kim, Y., Kim, U.M. A Development of Power Consumption Measurement System for Android Smartphones. Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication. 2017, p. 1–4. DOI: 10.1145/3022227.3022321.
12. Sankaranarayanan, Y., Ravindran, S., Ahamed, S., Balendraraja, K. Malware Sandbox Evasion Techniques in Mobile Devices. International Journal of Engineering and Management Research. 2022, 12(6), p. 69–74. DOI: 10.31033/ijemr.12.6.9.
13. Li, W., Xia, Y., Chen, H. Research on ARM TrustZone. GetMobile: Mobile Computing and Communications. 2019, 22(3), p. 17–22. URL: https://www.sigmobile.org/pubs/getmobile/articles/Vol22Issue3_1.pdf (дата обращения: 10.04.2025).
14. Waheed, M., Qadir, S. Effective and Efficient Android Malware Detection and Category Classification Using the Enhanced KronoDroid Dataset. Security and Communication Networks. 2024. DOI: 10.1155/2024/7382302.
15. Dash, S., Suarez-Tangil, G., Khan, S., Tam, K., Ahmadi, M., Kinder, J., Cavallaro, L. DroidScribe: Classifying Android Malware Based on Runtime Behavior. IEEE Security and Privacy Workshops. 2016. DOI: 10.1109/SPW.2016.25.
16. Zhu, H., Wei, H., Wang, L., Xu, Z., Sheng, V.S. An Effective End-to-End Android Malware Detection Method. Expert Systems with Applications. 2023, v. 218. DOI: 10.1016/j.eswa.2023.119593.
17. Thangavelooa, R., Jinga, W.W., Lenga, C.K., Abdullaha, J. Datdroid: Dynamic Analysis Technique in Android Malware Detection. International Journal on Advanced Science, Engineering and Information Technology. 2020, 10(2), p. 536–541. DOI: 10.18517/ijaseit.10.2.10238.
18. Singh, A., Tanha, M., Girdhar, Y., Hunter, A. Interpretable Android Malware Detection Based on Dynamic Analysis. ICISSP. 2024, p. 195–202. URL: https://www.scitepress.org/Papers/2024/124158/124158.pdf (дата обращения: 10.04.2025).
19. Faghihi, F., Zulkernine, M., Ding, S. CamoDroid: An Android Application Analysis Environment Resilient Against Sandbox Evasion. Journal of Systems Architecture. 2022, v. 125. DOI: 10.1016/j.sysarc.2022.102452.
20. Qadri, J., Chen, T.M., Blasco, J. A Review of Significance of Energy-Consumption Anomaly in Malware Detection in Mobile Devices. International Journal of Cyber Situational Awareness. 2016, 1(1), p. 210–230. DOI: 10.22619/IJCSA.2016.1001010.
21. Surendran, R., Uddin, M.M., Thomas, T., Pradeep, G. Android Malware Detection Based on Informative Syscall Subsequences. IEEE Access. 2024. DOI: 10.1109/ACCESS.2024.3387475.

REFERENCES:

[1] Massarelli, L., Aniello, L., Ciccotelli, C., Querzoni, L., Ucci, D., Baldoni, R. AndroDFA: Android Malware Classification Based on Resource Consumption. Information. 2020. 11(6), 326. DOI: 10.48550/arXiv.1709.00875.
[2] Dashevskyi, S., Zhauniarovich, Y., Gadyatskaya, O., Pilgun, A., Ouhssain, H. Dissecting Android Cryptocurrency Miners. Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy. 2020, p. 191–202. DOI: 10.48550/arXiv.1905.02602.

[3] Zhi, L.H., Selvarajah, V. Android Phone Hacking and Impact via Androrat. AIP Conference Proceedings. 2024, v. 2802, no. 1. DOI: 10.1063/5.0181916.
[4] Suresh, S., Di Troia, F., Potika, K. et al. An analysis of Android adware. J Comput Virol Hack Tech 15, p. 147–160 (2019). DOI: https://doi.org/10.1007/s11416-018-0328-8.
[5] PranavRaj, R., Syam, S.R. Identification Of Malware Families Using Energy Consumption. 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). 2024, p. 1–6. DOI: 10.1109/ICCCNT61001.2024.10724670.
[6] Shi, C., Cheng, C.C.C., Guan, Y. Forensic Analysis on Joker Family Android Malware. 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). 2021, p. 403–406. DOI: 10.1109/WiMob52687.2021.9606310.
[7] Rahman, M., Rahman, M., Carbunar, B., Chau, D.H. Search Rank Fraud and Malware Detection in Google Play. IEEE Transactions on Knowledge and Data Engineering. 2017, 29(6), p. 1329–1342. DOI: 10.1109/TKDE.2017.2667658.
[8] Kumar, K.A., Raman, A., Gupta, C., Pillai, R.R. The Recent Trends in Malware Evolution, Detection and Analysis for Android Devices. Journal of Engineering Science & Technology Review. 2020, 13(4). DOI: 10.25103/jestr.134.25.
[9] Arshad, S., Shah, M.A., Khan, A., Ahmed, M. Android Malware Detection & Protection: A Survey. International Journal of Advanced Computer Science and Applications. 2016. 7(2). DOI: 10.14569/IJACSA.2016.070262.
[10] Yalew, S.D. Mobile Device Security with ARM TrustZone. KTH Royal Institute of Technology, 2018. URL: https://www.semanticscholar.org/paper/Mobile-Device-Security-with-ARM-TrustZone-Yalew/d39f714e82724fb1d2352daf8e3814ee94fd8851 (accessed: 10.04.2025).
[11] Chung, J., Jung, H., Koo, J., Kim, Y., Kim, U.M. A Development of Power Consumption Measurement System for Android Smartphones. Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication. 2017, p. 1–4. DOI: 10.1145/3022227.3022321.
[12] Sankaranarayanan, Y., Ravindran, S., Ahamed, S., Balendraraja, K. Malware Sandbox Evasion Techniques in Mobile Devices. International Journal of Engineering and Management Research. 2022, 12(6), p. 69–74. DOI: 10.31033/ijemr.12.6.9.
[13] Li, W., Xia, Y., Chen, H. Research on ARM TrustZone. GetMobile: Mobile Computing and Communications. 2019, 22(3), p. 17–22. URL: https://www.sigmobile.org/pubs/getmobile/articles/Vol22Issue3_1.pdf (accessed: 10.04.2025).
[14] Waheed, M., Qadir, S. Effective and Efficient Android Malware Detection and Category Classification Using the Enhanced KronoDroid Dataset. Security and Communication Networks. 2024. DOI: 10.1155/2024/7382302.
[15] Dash, S., Suarez-Tangil, G., Khan, S., Tam, K., Ahmadi, M., Kinder, J., Cavallaro, L. DroidScribe: Classifying Android Malware Based on Runtime Behavior. IEEE Security and Privacy Workshops. 2016. DOI: 10.1109/SPW.2016.25.
[16] Zhu, H., Wei, H., Wang, L., Xu, Z., Sheng, V.S. An Effective End-to-End Android Malware Detection Method. Expert Systems with Applications. 2023, v. 218. DOI: 10.1016/j.eswa.2023.119593.
[17] Thangavelooa, R., Jinga, W.W., Lenga, C.K., Abdullaha, J. Datdroid: Dynamic Analysis Technique in Android Malware Detection. International Journal on Advanced Science, Engineering and Information Technology. 2020, 10(2), p. 536–541. DOI: 10.18517/ijaseit.10.2.10238.
[18] Singh, A., Tanha, M., Girdhar, Y., Hunter, A. Interpretable Android Malware Detection Based on Dynamic Analysis. ICISSP. 2024, p. 195–202. URL: https://www.scitepress.org/Papers/2024/124158/124158.pdf (accessed: 10.04.2025).
[19] Faghihi, F., Zulkernine, M., Ding, S. CamoDroid: An Android Application Analysis Environment Resilient Against Sandbox Evasion. Journal of Systems Architecture. 2022, v. 125. DOI: 10.1016/j.sysarc.2022.102452.
[20] Qadri, J., Chen, T.M., Blasco, J. A Review of Significance of Energy-Consumption Anomaly in Malware Detection in Mobile Devices. International Journal of Cyber Situational Awareness. 2016, 1(1), p. 210–230. DOI:10.22619/IJCSA.2016.1001010.
[21] Surendran, R., Uddin, M.M., Thomas, T., Pradeep, G. Android Malware Detection Based on Informative Syscall Subsequences. IEEE Access. 2024. DOI: 10.1109/ACCESS.2024.3387475.

Статья поступила в редакцию 16.04.2025; одобрена после рецензирования 06.06.2025; принята к публикации 07.07.2025
The article was submitted 16.04.2025; approved after reviewing 06.06.2025; accepted for publication 07.07.2025