МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

АВТОМАТИЧЕСКОЕ ВЫЯВЛЕНИЕ УЯЗВИМОСТЕЙ В МОБИЛЬНЫХ ПРИЛОЖЕНИЯХ НА ОСНОВЕ АНАЛИЗА БАЙТ-КОДА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Бикбулатова Артура Каригатовича

Научный руководитель
Профессор _____ Л. В. Кальянов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2025

ВВЕДЕНИЕ

Мобильные приложения активно используются для обработки конфиденциальных данных, что делает их привлекательной мишенью для злоумышленников. Существующие угрозы безопасности мобильных приложений (небезопасное хранение данных, криптография, некорректные конфигурации и др.) требуют эффективных методов выявления. Ручной анализ большого объема кода неэффективен. Автоматизированный статический анализ байт-кода является необходимым инструментом для масштабируемого обнаружения уязвимостей на ранних этапах разработки.

Целью является разработка десктопного приложения для автоматизированного статического анализа мобильных приложений на платформе Android с целью выявления потенциальных уязвимостей (Hardcoded Secrets, Небезопасная криптография, Небезопасное хранение данных) на основе анализа манифеста и байт-кода.

Задачи:

— Исследовать угрозы безопасности и методы анализа мобильных приложений.
— Изучить структуру APK, манифеста и байт-кода Dalvik/ART как объектов статического анализа.
— Выбрать и формализовать типы уязвимостей для детектирования на основе паттернов и эвристик.
— Разработать/адаптировать алгоритмы детектирования уязвимостей методом статического анализа байт-кода и манифеста.
— Разработать программный инструмент для комплексного статического анализа.
— Разработать прототип инструмента с графическим интерфейсом (GUI).
— Реализовать автономный запуск приложения путем упаковки.
— Провести демонстрацию работы инструмента на наборе мобильных приложений.
— Провести экспериментальное исследование инструмента.
— Проанализировать работу детекторов, возможности и ограничения.

Объектом исследования являются мобильные приложения для платформы Android (файлы APK). Предметом исследования являются методы и средства автоматического статического анализа для выявления уязвимостей

в мобильных приложениях, фокусируясь на анализе байт-кода Dalvik/ART и AndroidManifest.xml. Использованы методы анализа литературы, системного анализа, статического анализа кода, экспериментального исследования и сравнительного анализа. Результатом является инструмент для разработчиков и специалистов ИБ.

Выпускная квалификационная работа состоит из введения, четырех глав, заключения, списка использованных источников и приложений. Главы называются: "Основы безопасности мобильных приложений и статического анализа байт-кода "Методология выявления уязвимостей на основе статического анализа байт-кода "Программная реализация инструмента выявления уязвимостей "Демонстрация возможностей инструмента и сравнительный анализ с аналогами".

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе

В этом разделе представлены теоретические основы безопасности мобильных Android-приложений, подробно описана структура APK и байт-кода Dalvik/ART, рассмотрены актуальные уязвимости, поддающиеся статическому анализу, методы статического анализа байт-кода и обзор существующих инструментов.

— Представлены теоретические основы, необходимые для понимания предметной области анализа безопасности мобильных приложений и применяемых методов исследования.

— Детально описана структура APK-файлов, являющихся установочными пакетами мобильных приложений для платформы Android. Рассмотрены их ключевые составные части, такие как файл AndroidManifest.xml, содержащий декларативную информацию о приложении, его компонентах и разрешениях, а также байт-код Dalvik/ART (DEX-файлы), представляющий скомпилированный исполняемый код приложения. Подчеркнута роль этих файлов как основных объектов для статического анализа безопасности.

— Проведен анализ актуальных угроз и уязвимостей безопасности мобильных приложений, представляющих значительные риски для пользователей и данных. Особое внимание уделено тем типам уязвимостей, которые могут быть эффективно выявлены с помощью статического анализа байт-кода, включая: Hardcoded Secrets (жестко закодированные секреты в коде), уязвимости, связанные с Использованием небезопасной криптографии (слабые алгоритмы, некорректные режимы), и Небезопасное хранение данных на локальном устройстве.

— Описаны методы автоматизированного анализа безопасности мобильных приложений (включая ручной, динамический, статический и гибридный анализ). Особый акцент сделан на преимуществах статического анализа для систематического и полного (в теории) исследования кода приложения без его фактического выполнения, что позволяет выявлять уязвимости на уровне кода.

— Представлены базовые техники статического анализа байт-кода DEX. Описан парсинг формата DEX, позволяющий извлекать и структури-

ровать информацию о классах, методах и инструкциях. Рассмотрены методы анализа потока управления (Control Flow Analysis - CFA) для построения графов выполнения и анализа потока данных (Data Flow Analysis - DFA) для отслеживания распространения данных. Также описана техника поиска по паттернам инструкций как метод прямого поиска специфических конструкций кода, являющихся индикаторами уязвимостей.

— Раздел завершается обзором существующих инструментов статического анализа Android-приложений. Выделены их основные возможности, применяемые подходы к анализу и общая применимость для решения различных задач безопасности.

Во втором разделе

В данном разделе изложена разработанная методология выявления выбранных типов уязвимостей (Hardcoded Secrets, Небезопасная криптография, Небезопасное хранение данных) на основе статического анализа байт-кода Dalvik/ART с формализацией критериев детектирования в виде паттернов инструкций и эвристик,

— Изложена методология выявления потенциальных уязвимостей, специфичных для мобильных приложений на платформе Android, основанная на применении методов статического анализа непосредственно к байт-коду Dalvik/ART, который содержится в DEX-файлах APK-архива.

— Для каждого из трех выбранных типов уязвимостей (Hardcoded Secrets, Использование небезопасной криптографии, Небезопасное хранение данных) формализованы конкретные и операциональные критерии детектирования. Эти критерии выражены в виде характерных паттернов инструкций байт-кода Dalvik/ART и набора эвристических правил, которые применяются либо к данным, обрабатываемым этими инструкциями, либо к сигнатурам вызываемых методов API.

— Подробно описан подход к детектированию Hardcoded Secrets. Этот метод заключается в поиске специфических инструкций байт-кода, отвечающих за загрузку строковых констант в регистры (в частности, инструкция const-string). После обнаружения таких инструкций применяются разработанные эвристики к содержимому загружаемых строк. Эти эвристики могут включать поиск по предопределенным ключевым

словам (например, "password "api_key"), проверку длины строки (например, минимальная длина для секретов) или анализ на соответствие определенным форматам данных/путей.

— Для выявления уязвимостей, связанных с Небезопасной криптографией и Небезопасным хранением данных, используется подход, основанный на анализе вызовов методов API. Инструмент ищет инструкции вызова методов (invoke-* различных типов, таких как invoke-virtual, invoke-static и др.). Затем сигнатуры обнаруженных вызываемых методов (включая полное имя класса, имя метода и дескриптор параметров/возвращаемого типа) сравниваются с предопределенными списками. Для криптографии используется список "опасных"API, содержащий сигнатуры методов, часто используемых в небезопасных сценариях (например, вызовы getInstance для слабых алгоритмов). Для небезопасного хранения данных используется список "Sink-методов содержащий сигнатуры методов, предназначенных для записи данных в локальное хранилище (например, методы работы с SharedPreferences или файлами).

— Обоснован выбор языка программирования Python и фреймворка Androguard как основной технологической основы для реализации программного инструмента. Этот выбор мотивирован высокой пригодностью Python для задач анализа данных и автоматизации, а также широкими возможностями Androguard для парсинга форматов APK и DEX, детального низкоуровневого анализа байт-кода Dalvik/ART, включая доступ к инструкциям, строковым константам и информации о методах, что критически важно для реализации разработанной методологии.

— Представлена общая модульная архитектура разработанного программного инструмента. Описано взаимодействие ключевых компонентов, выполняющих последовательные этапы анализа: модуля парсинга и загрузки, отвечающего за извлечение и базовый анализ APK/DEX/Manifest; модуля обхода структуры кода, обеспечивающего систематическую итерацию по всем методам, базовым блокам и инструкциям байт-кода; модуля детекторов уязвимостей, содержащего логику для идентификации каждого конкретного типа уязвимости на основе формализованных патернов и эвристик; и модуля сбора и отчетности результатов, предна-

значенного для агрегации найденных уязвимостей и их представления пользователю в удобном формате.

В третьем разделе

Этот раздел посвещен описанию программной реализация инструмента, включая аналитическое ядро на Python с Androguard, разработке графического интерфейса на Tkinter/ttk, механизма запуска анализа в отдельном потоке и процессу упаковки приложения с помощью PyInstaller.

— Описана программная реализация разработанного инструмента для автоматизированного выявления уязвимостей в мобильных приложениях на платформе Android.

— Детализирована реализация основного аналитического модуля, который составляет ядро инструмента. Этот модуль разработан на языке Python и активно использует богатый функционал фреймворка Androguard. Описано, как Androguard применяется для парсинга APK-файлов, получения структурированного доступа к содержимому DEX-файлов, и выполнения итерации по классам, методам, базовым блокам и отдельным инструкциям байт-кода, что является основой для низкоуровневого анализа.

— Представлена программная реализация детекторов для каждого из трех выбранных типов уязвимостей (Hardcoded Secrets, Небезопасная криптография, Небезопасное хранение данных). Описано, как эти детекторы реализованы в коде на основе разработанных паттернов инструкций байт-кода и эвристических правил, применяемых для идентификации потенциально опасных конструкций и вызовов методов.

— Подробно описана разработка графического пользовательского интерфейса (GUI), который обеспечивает удобное взаимодействие пользователя с аналитическим ядром. GUI реализован с использованием стандартной библиотеки Python Tkinter и ее расширения ttk для создания более современного внешнего вида. Описана общая структура интерфейса, размещение различных виджетов (таких как поля ввода/вывода для пути к файлу и результатов, кнопки для выбора файла и запуска анализа, прогресс-бар для отображения хода выполнения) и реализация логики взаимодействия с пользователем (обработка событий выбора файла, запуск процесса анализа).

— Объяснен механизм запуска ресурсоемкого процесса анализа в отдельном потоке выполнения с использованием стандартного модуля Python threading. Этот подход применен для предотвращения блокировки основного потока, ответственного за отрисовку и обработку событий GUI, тем самым обеспечивая отзывчивость интерфейса во время длительного анализа. Описан способ обмена информацией о текущем прогрессе анализа и его финальных результатах между рабочим и основным потоками.
— Рассмотрена задача упаковки разработанных модулей (аналитического ядра и GUI) в автономное десктопное приложение, не требующее установки зависимостей Python у конечного пользователя. Для этого был выбран инструмент PyInstaller. Описаны сложности, возникшие в процессе упаковки, в частности, связанные с корректным включением скрытых зависимостей Androguard и обработкой файловых путей внутри упакованного приложения. Представлены предложенные решения, позволившие преодолеть эти трудности и успешно создать исполняемый файл.

В четвертом разделе

В данном разделе представлены результаты демонстрации работы инструмента на реальных приложениях с примерами вывода (GUI, JSON) и проведен сравнительный анализ разработанного прототипа с аналогами, выделив его ключевые преимущества

— Представлены результаты демонстрации возможностей разработанного программного инструмента и проведен сравнительный анализ с существующими аналогами на рынке средств анализа безопасности мобильных приложений.
— Описан тестовый набор, сформированный специально для проведения демонстрации работы инструмента и последующего экспериментального исследования. Этот набор включал реальные популярные мобильные приложения-мессенджеры (такие как Telegram, Яндекс Мессенджер, VK Мессенджер), выбранные за их широкую распространенность, актуальность вопросов безопасности и значительный объем кода для анализа.
— Представлены конкретные примеры вывода результатов анализа, полу-

ченных разработанным инструментом при обработке приложений из тестового набора. Показано, как обнаруженные потенциальные уязвимости выбранных типов отображаются как в графическом пользовательском интерфейсе (GUI) инструмента, так и в структурированном формате JSON, наглядно иллюстрируя процесс детектирования Hardcoded Secrets, Небезопасной криптографии и Небезопасного хранения данных.

— Проведен сравнительный анализ разработанного инструмента-прототипа с существующими на рынке аналогами для статического анализа безопасности Android-приложений (включая такие как MobSF и другие).

— В результате сравнительного анализа выделены ключевые преимущества разработанного локального некоммерческого решения. К этим преимуществам относятся: обеспечение конфиденциальности анализируемых данных и кода приложения (поскольку анализ полностью происходит на локальной машине пользователя без передачи данных сторонним сервисам); открытость и гибкость кода инструмента, позволяющие его детальное изучение, модификацию и адаптацию под специфические задачи или добавление новых детекторов; а также удобство формата JSON-отчета для дальнейшей автоматизированной обработки результатов и их легкой интеграции в другие системы и процессы безопасности.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной дипломной работы была успешно решена поставленная задача по исследованию и разработке прототипа десктопного инструмента для автоматизированного статического анализа мобильных приложений на платформе Android. Цель, заключающаяся в выявлении потенциальных уязвимостей типа Hardcoded Secrets, Небезопасной криптографии и Небезопасного хранения данных на основе анализа байт-кода Dalvik/ART, достигнута.

В процессе работы проведен анализ предметной области, изучена структура APK-файлов и байт-кода, формализованы паттерны выбранных уязвимостей. Разработан и реализован модуль статического анализа на Python с использованием фреймворка Androguard, включающий детекторы уязвимостей. Создан графический пользовательский интерфейс на базе Tkinter для удобства взаимодействия. Выполнена упаковка приложения в автономный исполняемый файл с помощью PyInstaller.

Демонстрационное исследование на реальных мобильных приложениях подтвердило работоспособность инструмента и его способность выявлять потенциальные уязвимости выбранных типов, основываясь на анализе инструкций и вызовов методов байт-кода. Сравнительный анализ с существующими аналогами выявил ключевые преимущества разработанного прототипа, такие как обеспечение конфиденциальности данных за счет локального выполнения, открытость и прозрачность кода инструмента, а также удобство формата JSON для автоматизированной обработки результатов, при этом обозначив ограничения базового уровня применяемых аналитических техник.

Таким образом, разработанный инструмент является рабочим прототипом, демонстрирующим принципы статического детектирования уязвимостей на уровне байт-кода Android-приложений, и послужит основой для возможного дальнейшего развития.

Основные источники информации:

1 Документация Android Developer[Электронный ресурс] URL: https://developer.android.com/studio/intro?hl=ru

2 Android – Static Analysis [Электронный ресурс] URL: https://sweet.ua.pt/jpbarraca/course/er/slides/er-3-android_static_analysis.pdf

3 Документация фреймворка Androguard [Электронный ресурс] URL: https://

androguard.readthedocs.io/en/latest/

4 Zuddas, S., Marchesini, J., & Miori, R. (2019). Mobile malware static analysis: A comparative study. В Proceedings of the 14th International Conference on Availability, Reliability and Security [Электронный ресурс] URL: https://androguard.readthedocs.io/en/latest/ (дата обращения: 11.04.2

5 Nikolay Elenkov. "Android Security Internals: An In-Depth Guide to Android's Security Architecture (2014)"

6 Android Application Security - Securing Android Apps for Developers [Электронный ресурс] URL: https://snyk.io/articles/application-security/mobile-application-security/android-application-security/

7 Ren, X., Liu, Y., Li, S., Xing, X., & Kim, T. (2017, August). Cryptoguard: A tool for detecting cryptographic misuses in android applications. В 2017 26th USENIX Security Symposium (USENIX Security 17) (pp. 1013-1029).

8 Guide to Android Application Security [Электронный ресурс] URL: https://digital.ai/catalyst-blog/android-app-security/

9 Rasthofer, S., Arzt, S., Milani Fard, A., Bodden, E., & Le Traon, Y. (2014). DroidBench: a benchmark suite for security analysis of android applications. International Journal on Software Tools and Technology Transfer, 16(4), 421-436.