Инженерный вестник Дона, №7 (2025)
ivdon.ru/ru/magazine/archive/n7y2025/10238

Безопасность Android-приложений

С.В. Нечаев[1], И.Ф. Занин[1], Н.В. Беспалова[2], А.С. Ершов[3], Н.Ю. Хороводова[3]

[1]Национальный исследовательский университет ИТМО, Санкт-Петербург
[2]Финансовый университет при Правительстве Российской Федерации, Москва
[3]Саратовский государственный технический университет им. Гагарина Ю.А., Саратов

Аннотация: В работе предлагается разработка собственного решения, позволяющего увеличить точность и полноту обнаружения уязвимостей в клиентской части Android-приложений. В ходе работы были определены основные актуальные угрозы для Android-приложений. В качестве основного метода исследования приложений был выбран метод динамического анализа, позволяющего провести тестирование безопасности приложения во время его работы, тем самым имитируя действия злоумышленника. Был разработан сканер обнаружения уязвимостей в клиентской части Android-приложений, который основывался как на динамическом, так и на статическом анализе. Приведены результаты полного тестирования разработанного программного обеспечения, в ходе которого детектированы различные проблемы безопасности. Разработанное программное обеспечение может быть внедрено в процессы жизненного цикла безопасной разработки программного обеспечения организации, для повышения уровня обеспечения целостности, конфиденциальности и доступности данных пользователя, хранимых Android-приложением. Проведенный анализ позволяет сделать выводы о наличии достаточно большого числа проблем безопасности в Android-приложениях, что может свидетельствовать об их низком уровне защищенности и как следствие, необходимости формирования комплексного решения для обеспечения безопасности Android-приложений, включающего в себя применение практик безопасной разработки и регулярных проверок безопасности.
Ключевые слова: безопасность Android-приложений, динамический анализ, статический анализ, обнаружение уязвимостей, сканер безопасности.

Введение

Android является самой популярной мобильной операционной системой в мире. Согласно данным платформы StatCounter, по состоянию на ноябрь 2024 года, во всем мире доля Android на рынке составляла 72% среди пользователей смартфонов, по сравнению с долей рынка iPhone в 28%. Под Android написано множество приложений, которые плотно вошли в жизнь современного человека. Эти приложения хранят, обрабатывают, передают конфиденциальную информацию. Эксплуатация программных уязвимостей в

Android-приложениях может повлечь за собой серьезные последствия, что делает обеспечение безопасности первостепенной задачей [1,2].

Количество проблем безопасности в Android приложениях постоянно растет: согласно данным организации “Информзащита”, в 2024 году число атак на API (Application Programming Interface) мобильных приложений в России увеличилось на 630 % по сравнению с прошлым годом [3,4].

Для обеспечения безопасности Android приложений существуют различные подходы, позволяющие сократить число проблем безопасности в разработанном приложении до того момента, как оно будет доставлено конечному пользователю. Все проверки безопасности, используемые для исследования уже разработанного приложения можно разделить на ручные и автоматические. Для ручного тестирования безопасности необходимы специалисты, которые на регулярной основе будут заниматься исследованиями приложения. Для компаний, обладающих малым числом сотрудников, наиболее оптимальным способом является внедрение автоматических проверок.

В данной работе предлагается подход к разработке решения, позволяющего увеличить точность и полноту обнаружения уязвимостей в клиентской части Android-приложений с помощью метода динамического анализа.

Актуальные угрозы для Android-приложений

Для мобильных приложений характерна более обширная поверхность атаки, чем для их веб-аналогов, поскольку они скачиваются с публичных площадок и дают возможность проинспектировать код, кроме этого пользовательская информация, которая содержится в них является крайне привлекательной для киберприступников [5,6]. Определение актуальных угроз является основой разработки мобильного приложения в защищенном

исполнении, в таблице 1. представлен ряд актуальных угроз и возможные решения по их предотвращению [7,8]:

Таблица 1 - Актуальные угрозы и возможные решения по их предотвращению

№	Наименование угрозы	Потенциальные проблемы	Возможность предотвращения
1	**Неправильное использование платформы**	Наиболее частый вектор атаки для подобных проблем - с помощью методов социальной инженерии заставить пользователя установить на свое устройство вредоносное приложение, которое может маскироваться под легитимное.	• применения правильных настроек сервера; • ограничение доступа к приложению, реализация ограничительных прав доступа к файлам.
2	**Небезопасное хранение данных**	Доступ и извлечение информации из данных, которые хранятся во внешней памяти. Доступ к информации, хранимой во внутренней памяти, которая реализуется через специальную компоненту приложения - поставщик контента.	• тестирование приложения на уязвимость угрозам и определение механизмов взаимодействия обрабатываемых данных с API.
3	**Небезопасная коммуникация**	Злоумышленник, который находится в одной сети с жертвой, или установивший на устройство жертвы вредоносное приложение, сможет получить доступ к	• установка SSL/TLS сертификатов от проверенных центров сертификации. • SSL-пиннинг.

№	Наименование угрозы	Потенциальные проблемы	Возможность предотвращения
		передаваемым данным.	
4	**Небезопасная аутентификация**	Обход аутентификации обычно реализуется через существующие уязвимости, такие как неправильная проверка сервисных запросов со стороны сервера.	• Исключить локальную аутентификацию. • Использовать многофакторную аутентификацию.
5	**Недостаточная криптографическая стойкость**	Для хеширования паролей в локальных базах данных SQLite разработчики часто используют MD5 хеши. Данный алгоритм не является криптостойким. При шифровании различных файлов могут быть использованы блочные шифры в режиме электронной кодовой книги (ECB), что сильно может упростить процесс криптоанализа злоумышленнику.	• Применять строгие стандарты криптографии
6	**Небезопасная авторизация**	Слабые схемы авторизации, несмотря на успешную проверку подлинности пользователя, могут не справляться с проверкой его прав на доступ к запрашиваемым ресурсам. Подобный недочет позволяет	• Проверять роли и разрешения аутентифицированного пользователя, используя информацию из бэкенд-систем, а не из мобильного устройства.

№	Наименование угрозы	Потенциальные проблемы	Возможность предотвращения
		хакерам авторизоваться и выполнять атаки с целью повышения привилегий.	
7	**Качество кода клиента**	При наличии ошибок в коде клиентской части Android-приложений атакующий может передавать особые входные данные в вызовы функций, провоцируя их выполнение и наблюдая за поведением приложения.	• Использовать автоматизированные инструменты для тестирования буфера на переполнение, определения утечек памяти • Применять в организации согласованные шаблоны написания кода.
8	**Подделка кода**	Заставить жертву установить вредоносное приложение, которое имитирует легитимное. В такое приложение встраивается вредоносный код, с помощью которого пользователь может быть атакован.	• внедрить техники, препятствующие выполнению поддельных приложений, например, контрольные суммы, цифровые подписи, усиление кода и прочие методы проверки.
9	**Реверс-инжиниринг**	Злоумышленники могут декомпилировать приложение для изучения принципов работы.	• Применять обфускацию
10	**Лишняя функциональность**	Злоумышленники изучают файлы конфигурации, двоичные файлы и прочие компоненты, раскрывая функционал бэкенд-части, который затем используют	• проверить настройки приложения на предмет наличия скрытых переключателей. • Убедиться, что логи не содержат чрезмерно описательных

№	Наименование угрозы	Потенциальные проблемы	Возможность предотвращения
		для совершения атак.	инструкций о работе сервера.

Системы обнаружения уязвимостей в клиентской части Android-приложений

Оценку безопасности приложения позволяет провести анализ его кода. Существует множество классификаций анализа, но чаще всего в большинстве организацией применяются два типа анализа:

- Статический;
- Динамический.

В ходе работы был разработан сканер обнаружения уязвимостей в клиентской части Android-приложений, который основывается как на динамическом, так и на статическом анализе, поскольку до динамического анализа необходимо, проанализировав исходный код, выявить точки входа [9, 10]. В качестве метода обнаружения точек входа предлагается использовать регулярные выражения. С помощью них происходит поиск необходимых данных в исходном коде приложения, написанном на Java.

Разработанный сканер состоит из следующих модулей:

- Модуль исследования исходного кода кода. Данный модуль осуществляет получение имен и путей хранения всех файлов проекта, поиск точек входа в приложение, поиск частей кода, отвечающих за отправку intent-ов к компонентам приложения, а также регистрация обработчиков URI-схем, по которым может происходить обращение к поставщику контент
- Модуль взаимодействия с Android-девайсом. Данный модуль предназначен для автоматизации выполнения следующих операций: установка приложения на устройство, запуск приложения, отправка intent-ов к компонентам приложения, получение данных через поставщик контента

- Модуль, реализующий динамический анализ. Данный модуль реализует механизм динамического анализа, отправляя к активности уязвимого приложения, содержащей Android WebView intent, в котором передается полезная нагрузка, эксплуатирующая уязвимость.
- Модуль генерации отчетов. Предназначен для представления результата сканирования, содержащего в себе подробное описание найденных уязвимостей

Список уязвимостей, которые может обнаруживать сканер безопасности, а также основные методы их выявления представлены в Таблице 2:

Таблица 2 - Список уязвимостей и основные методы их выявления

Компонента	**Название уязвимости**	**Метод обнаружения**
Активность	XSS в Android WebView	Внедрение на страницу JavaScript-кода, отправляющего запрос на подконтрольный сервер, расположенный в локальной сети сканируемого устройства
	Инъекция в loadUrl()	Внедрение в функцию URL-адреса страницы, расположенной на подконтрольном сервере
Поставщик контента	SQL-инъекция	Изменение запроса к базе данных с целью вызвать ошибку при его выполнении или корректное выполнение запроса с функциональным изменением
	Path Traversal	Подмена получаемого файла на файл /etc/hosts с итеративным поднятием по директориям на

Компонента	Название уязвимости	Метод обнаружения
		Android-девайсе
Сервис	Незащищённое копирование на SD-карту	Подмена имени копируемого файла на словарные имена различных приватных файлов. Мониторинг состояния файловой системы Android-девайса

Результаты тестирования разработанного решения

Для процесса тестирования и исследования эффективности разработанного решения были выбраны следующие Android приложения, содержащие в себе различные уязвимости:

- Super Secure App - тестовое приложение, разработанное для отладки сканера
- NSA - приложение, используемое для обучения мобильной безопасности
- Sieve - уязвимое приложение, доступное на GitHub
- Vulnerable WebView - уязвимое приложение, доступное на GitHub

В результате сканирования были обнаружены уязвимости, например, на рисунке 1 представлен фрагмент отчета, содержащий в себе информацию об обнаруженной инъекции в функцию loadUrl ():

```
==================
=     Results     =
==================
[1] URL Injection has been detected
Declaration of intent  Intent intent = new Intent(this, BrowserActivity.class);
An intent was sent to "BrowserActivity" and contains a vulnerable parameter "url"
What happened during the scan:
The server received a request from webview
You can exploit it with payload: https://cdn.iconscout.com/icon/premium/png-256-thumb/computer-hacked-6193610-5152893.png
ADB command:
adb shell am start -n ru.startandroid.supersecureapp/.BrowserActivity -e url "https://cdn.iconscout.com/icon/premium/png-256-thumb/compute
r-hacked-6193610-5152893.png"
file: D:\Диплом\Dast\test\SuperSecureApp\app\src\main\java\ru\startandroid\supersecureapp\InternetActivity.java
==================
```

Рисунок 1 - Сообщение об обнаруженной инъекции в loadURL()

Следующая проблема безопасности, обнаруженная сканером, - уязвимость межсайтового скриптинга. Информация из отчета о данном срабатывании представлена на Рисунке 2:

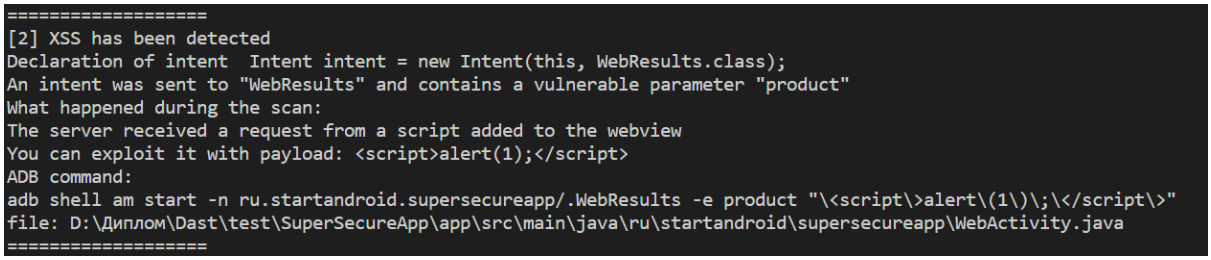

Рисунок 2 - Информация о выявленной XSS

В отчете указано, что срабатывание было зарегистрировано в связи получением запроса, отправленным с помощью JavaScript-кода, внедренного в Android WebView.

В результате полного тестирования разработанного программного обеспечения были выявлены различные проблемы безопасности в 4 приложениях. Результаты исследования безопасности данных приложений представлены в Таблице 3:

Таблица 3 – Исследование безопасности данных приложений

Приложение	**Обнаруженные проблемы безопасности**
Super Secure App	● XSS в Android WebView ● Инъекция в loadUrl() ● SQL-инъекция в поставщике контента ● Path Traversal в поставщике контента ● Безопасное копирование информации с использованием сервиса
NSA	● Инъекция в loadUrl(), приводящая к XSS в Android WebView
Sieve	● Path Traversal в поставщике контента
Vulnerable WebView	● Инъекция в loadUrl()

Для каждой из найденных уязвимостей были описаны действия, которые позволяют воспроизвести действия сканера для облегчения процесса валидации срабатываний.

Заключение

В представленной статье были определены основные проблемы безопасности, актуальные для Android-приложений, а также определены основные угрозы их безопасности. В качестве основного метода исследования приложений был выбран метод динамического анализа, позволяющего провести тестирование безопасности приложения во время его работы, тем самым имитируя действия злоумышленника. В результате была разработана методика тестирования безопасности Android-приложений.

В основе разработанной методики лежит динамический анализ, который опирается на обнаружение точек входа в исходном коде приложения. С использованием данной методики было разработано программное обеспечение, которое реализует процесс исследования Android-приложений, для которых доступен исходный код.

Проведенный анализ позволяет сделать выводы о наличии достаточно большого числа проблем безопасности в Android-приложениях, что может свидетельствовать об их низком уровне защищенности и как следствие, необходимости формирования комплексного решения для обеспечения безопасности Android-приложений. Такое комплексное решение должно включать в себя применение практик безопасной разработки и регулярных проверок безопасности, основанных как на методе статического, так и на методе динамического анализа.

Литература

1. Duo W., Zhou M. C., Abusorrah A. A survey of cyber attacks on cyber physical systems: Recent advances and challenges //IEEE/CAA Journal of Automatica Sinica. 2022. V. 9. №. 5. P. 784-800.

2. Ahmetoglu H., Das R. A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions // Internet of Things. 2022. V. 20. P. 100615.
3. Patterson C. M., Nurse J. R. C., Franqueira V. N. L. Learning from cyber security incidents: A systematic review and future research agenda //Computers & Security. 2023. V. 132. P. 103309.
4. Котенко И. В., Саенко И.Б., Паращук И.Б., Десницкий В.А., Виткова Л.А. Аналитическая обработка больших массивов данных о событиях кибербезопасности с применением суперкомпьютерных вычислений. // Программные продукты и системы. 2024. Т. 37. №. 4. С. 487–494.
5. Хромова А. Р., Петросян Л. Э. Анализ уязвимостей в системах безопасности данных // Инженерный вестник Дона. 2023. №. 6. URL: ivdon.ru/ru/magazine/archive/n6y2023/8447
6. Фатхи В. А., Дьяченко Н. В. Тестирование безопасности приложений // Инженерный вестник Дона. 2021. №. 5. URL: ivdon.ru/ru/magazine/archive/n5y2021/6947
7. Зегжда Д. П., Павленко Е. Ю. Анализ безопасности Android-приложений. Санкт-Петербург: Политех-пресс. 2021. 320с.
8. Изергин Д. А., Еремеев М. А., Магомедов Ш. Г., Смирнов С. И. Оценка уровня информационной безопасности мобильной операционной системы Android. // Российский технологический журнал. 2020. Т.7. № 6. С. 44-55.
9. Павленко Е. Ю., Игнатьев Г. Ю., Зегжда П. Д. Статический анализ безопасности Android-приложений // Проблемы информационной безопасности. Компьютерные системы. 2017. №. 4. С. 73-79.
10. Сковорода А. А., Гамаюнов Д. Ю. Динамический анализ мобильных приложений //Программная инженерия. 2019. Т. 10. №. 7-8. С. 324-333.

References

1. Duo W., Zhou M. C., Abusorrah A. IEEE/CAA Journal of Automatica Sinica. 2022. vol. 9. №. 5. pp. 784-800.
2. Ahmetoglu H., Das R. Internet of Things. 2022. Vol. 20. pp. 100615.
3. Patterson C. M., Nurse J. R. C., Franqueira V. N. L. Computers & Security. 2023. vol. 132. P. 103309.
4. Kotenko I. V., Saenko I.B., Parashhuk I.B., Desnickij V.A., Vitkova L.A. Programmnye produkty i sistemy. 2024. vol. 37. №. 4. pp. 487–494.
5. Hromova A. R., Petrosyan L. E. Inzhenernyj vestnik Dona. 2023. №. 6 URL: ivdon.ru/ru/magazine/archive/n6y2023/8447
6. Fathi V. A., D'yachenko N. V. Inzhenernyj vestnik Dona. 2021. №. 5. URL: ivdon.ru/ru/magazine/archive/n5y2021/6947
7. Zegzhda D. P., Pavlenko E. Ju. Analiz bezopasnosti Android-prilozhenij [Android Application Security Analysis]. Sankt-Peterburg: Politeh-press. 2021. 320 p.
8. Izergin D. A., Eremeev M. A., Magomedov Sh. G., Smirnov S. I. Rossijskij tehnologicheskij zhurnal. 2020. vol.7. № 6. pp. 44-55.
9. Pavlenko E. Ju., Ignat'ev G. Ju., Zegzhda P. D. Problemy informacionnoj bezopasnosti. Komp'juternye sistemy. 2017. №. 4. pp. 73-79.
10. Skovoroda A. A., Gamajunov D. Ju. Programmnaja inzhenerija. 2019. vol. 10. №. 7-8. pp. 324-333.

Дата поступления: 3.05.2025

Дата публикации: 27.06.2025