Санкт-Петербургский государственный университет

***ЗАЙЦЕВ Дмитрий Игоревич***

Отчет по производственной (преддипломной) практике

# Создание модуля повышения приватности пользователя приложений для Android устройств с несистемным административным доступом Magisk

Уровень образования: магистратура

Направление *02.04.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *ВМ.5665.2021 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
к. ф.-м. н., ст. преп., Ловягин Н. Ю.

Рецензент:
доцент ФГБОУ ВО РГПУ им. А. И. Герцена, к. ф.-м. н. Якубсон М. Я.

Санкт-Петербург
2023

Saint Petersburg State University

***ZAITCEV Dmitrii***

Master's Thesis

# Creation of a module for increasing user privacy of applications for Android devices with non-system administrative access Magisk

Education level: master

Speciality *02.04.03 «Software and Administration of Information Systems»*

Programme *BM.5665.2021 «Software and Administration of Information Systems»*

Scientific supervisor:
C.Sc., senior lecturer N.Y. Lovyagin

Reviewer:
docent at «The Herzen State Pedagogical University of Russia», C.Sc. M.Y. Yakubson

Saint Petersburg
2023

# Оглавление

# Введение

В настоящее время на рынке мобильных приложений присутствует большое количество программных продуктов. Только в первую половину 2020 года в Google Play было опубликовано 2,96 млн., а в AppStore 4,37 млн. приложений [10]. При этом данная статистика не учитывает приложения, которые не были опубликованы в официальных магазинах смартфонов.

Все эти приложения имеют самые разные функции и задачи. Однако из этого числа приложений встречается значительное количество вредоносного программного обеспечения, которое может навредить мобильному устройству пользователя или самому пользователю. Масштаб вреда может быть разным: например, спам, кража персональных данных, мошенничество и так далее. Только по данным Kaspersky Security Network в первый квартал 2021 года было обнаружено 1 451 660 вредоносных установочных пакетов [27]. Представленная статистика основана на детектирующих вердиктах продуктов «Лаборатории Касперского», которые были предоставлены пользователями, подтвердившими свое согласие на передачу статистических данных.

Одним из последствий ущерба является получение личной информации пользователя вредоносным приложением, которое обладает разрешением на доступ к приватным данным смартфона. В полученных вредоносным ПО данных могут оказаться контакты, аккаунты, геолокация и так далее, даже если они не нужны для выполнения функций приложения. Получение этих данных может привести к краже денег, контактов родственников, раскрытие информации о месте жительства пользователя, чем могут воспользоваться мошенники и преступники.

Для обеспечения защиты персональных данных с сохранением возможности корректной работы приложений необходима возможность выборочно подменять реальные данные случайными, пустыми, зашифрованными или специально заданными значениями. Это можно достигнуть, заменив системные библиотеки Android измененными версиями, что возможно при наличии административного доступа к устройству.

При этом данные версии могут быть как записаны непосредственно на системный раздел, так и быть предоставлены без вмешательства в него — например, с помощью модуля внесистемного (systemless) административного доступа Magisk [13] [26], подменяющего файлы при запуске устройства. Отсутствие изменения системного раздела `/system` делает возможным обновления-по воздуху (over-the-air). Также решение должно быть с открытым исходным кодом для гарантии безопасности пользователей самого приложения, скрывающего персональные данные.

На текущий момент уже существует несколько примеров программного обеспечения, которое позволяет подменять персональные данные. Например, модуль XPrivacy Lua [12] для фреймворка XPosed [17], Shizuku [18], LBE Privacy Guard [6] и PDroid Privacy Protection [7].

Для решения вышеописанной проблемы был выбран набор программного обеспечения для настройки Android — Magisk и включенный в него пакет Zygisk. Плюсом данного программного обеспечения являются отсутствие изменения системного раздела и открытость кода.

# 1. Постановка задачи

Целью работы является создание внесистемного модуля Magisk с открытым кодом, позволяющего выборочно подменять реальные данные во время обращения к ним вредоносных приложений.

Для её выполнения были поставлены следующие задачи:

- подготовить среду разработки и тестирования;
- выделить функции и библиотеки, отвечающие на запросы о персональных данных средствами Android API из их исходных кодов;
- создать модифицированные библиотеки, отвечающие за данные одного из выбранных запросов;
- локализовать собранные библиотеки;
- реализовать работающий прототип, модифицирующий стандартные библиотеки Android;
- реализовать Android приложение для выборочной работы подмены библиотек, устанавливающееся с Magisk-модулем;
- реализовать рабочий прототип с использованием Zygisk;
- проанализировать решения, использующие и не использующие Zygisk.

# 2. Обзор

## 2.1. Обзор существующих решений

В рамках данной работы были рассмотрены следующие варианты программного обеспечения: XPrivacy Lua, Shizuku, LBE Privacy Guard, PDroid Privacy Protection и Magisk, позволяющие вносить существенные изменения в работу мобильного устройства.

Функционал подмены пользовательских данных реализован в модуле XPrivacy Lua для фреймворка XPosed. Это платформа, которая позволяет приложениям изменять стандартную библиотеку Android. Однако часть функционала доступна только в платной версии. К тому же это отдельная тяжелая универсальная модульная система со слишком сильным вмешательством в системные библиотеки, что может вызвать проблемы с безопасностью и заблокировать телефон.

Shizuku дает возможность обычным приложениям использовать системные API напрямую с привилегиями `adb root` с Java процессом. Однако данное решение не является внесистемным.

Так же были рассмотрены альтернативные варианты: LBE Privacy Guard и PDroid Privacy Protection. Однако LBE Privacy Guard является уже устаревшим вариантом, который поддерживает Android, начиная с версии 2.0 и заканчивая версией 5.1. PDroid Privacy Protection также является устаревшими и работает с Android, начиная с версии 2.3.x и заканчивая версией 4.0.3, к тому же не является внесистемным решением.

Выбранное для решения данной проблемы программное обеспечение Magisk представляет собой бесплатное приложение с открытым исходным кодом, дающее административные права доступа, которые позволяют создавать модули, установка которых соответствует общей концепции приложения и происходит без модификации системного раздела. Такой подход позволяет в любой момент отключить все установленные модификации. Эти модули позволяют изменить системные настройки и работу мобильного устройства.

В 24 версии Magisk в утилиту был добавлен пакет Zygisk [25]. Он

позволяет разработчикам осуществлять запуск кода непосредственно в процессе каждого Android-приложения. Одним из основных недостатков данного решения является крайне ограниченное количество Zygisk-модулей и неинформативная документация. Однако данное решение считается более перспективным и современным.

Основываясь на перечисленном выше, было решено продолжить работу, использовав Magisk-модуль с использованием Zygisk и без него.

Поскольку нет гарантии, что установленное приложение не является вредоносным, в рамках проекта было решено считать любое установленное программное обеспечение вредоносным. В случае пользовательского доверия модуль можно будет выключить для конкретного приложения.

## 2.2. Используемые технологии

В рамках данного проекта были использованы следующие технологии:

- приложение с внесистемным административным доступом Magisk версий 23.0 и 25.2;
- виртуальная машина VirtualBox 6.1 [15] с установленными операционными системами Android 11.0 [3] и LineageOS 14.1 [11], которая является операционной системой с открытым кодом, основанной на ОС Android;
- Android Studio для эмуляции и тестирования модулей Magisk;
- LDPlayer — эмулятор для запуска Android-приложений;
- язык программирования Shell для написания модулей для Magisk;
- язык программирования Java для написания Android-библиотек;
- язык программирования C++ и JNI (Java Native Interface) для написания Zygisk-модуля;
- система управления базами данных SQLite для хранения данных Android-приложения.

## 2.3. Подготовка среды разработки и тестирования

В рамках данной работы было рассмотрено несколько вариантов установки программного обеспечения Magisk.

- На виртуальную машину с операционной системой Android и дистрибутивом Android — LineageOS. Однако из-за проблем с подключением `adb` установить права доступа администратора не удалось.
- На Android-эмулятор для игр LDPlayer [9]. Программное обеспечение Magisk установилось, однако возникла проблема, связанная с невозможностью тестирования и отладки модулей, поэтому данный вариант тоже не подошел.
- На эмулятор, который поставляется с Android Studio [19]. Magisk успешно установился, к тому же данная среда разработки позволяет достаточно удобно тестировать приложение.

Исходя из вышеописанных результатов, было решено использовать эмулятор Android Studio.

## 2.4. Структура модуля Magisk

В рамках данной работы, опираясь на руководство по разработке [14], был изучен процесс создания модуля. Он представляет собой `zip`-архив со следующей структурой:

```
/data/adb/modules
├── $MODID
│   ├── module.prop
│   ├── system
│   │   ├── ...
│   │   └── ...
│   ├── zygisk
│   │   ├── arm64-v8a.so
│   │   ├── armeabi-v7a.so
```

```
│   │   ├── x86.so
│   │   ├── x86_64.so
│   │   └── unloaded
│   ├── skip_mount
│   ├── disable
│   ├── remove
│   ├── customize.sh
│   ├── post-fs-data.sh
│   ├── service.sh
│   ├── uninstall.sh
│   ├── system.prop
│   ├── sepolicy.rule
│   ├── vendor
│   ├── product
│   ├── system_ext
│   └── ...
├── another_module
│   └── .
└── .
```

В рамках текущего проекта особое внимание уделялось файлам `module.prop`, `customize.sh`, а также директориям `/system` и `/zygisk`.

Файл `module.prop` содержит в себе информацию и метаданные о модуле. Например, название, описание, идентификатор и версию модуля, а также информацию об авторе.

Файл `customize.sh` содержит скрипт, который запускается при установке модуля.

Директория `/system` позволяет заменять файлы на устройстве. Она будет рекурсивно объединена с директорией `/system` устройства. Другими словами, существующие файлы из `/system` на устройстве будут заменены файлами из директории `/system` модуля, а файлы, которых еще нет в директории `/system` будут добавлены на устройство.

Директория `/zygisk` содержит собранные нативные библиотеки Zygisk-модуля.

## 2.5. Выделение библиотек

В рамках данного проекта было решено сначала добиться подмены контактов пользователя. Для этого был проанализирован процесс изменения файлов в программном обеспечении PDroid Privacy Protection. Процесс изменения системных библиотек [24] приведен для Android 2.3. Проведя анализ современных библиотек с помощью Google Git репозитория [8], было выяснено, что класс ProcessManager, который модифицировался приложением PDroid для перехвата обращения вредоносного ПО, был разбит на несколько файлов. Основная логика содержится в классах UNIXProcess и ProcessEnvironment. А изменяемый приложением PDroid статический класс ProcessImpl был вынесен в отдельный ProcessImpl класс с таким же названием.

Процесс создания библиотек для подмены пользовательских данных [16] был также изучен на примере изменения фреймворков стандартных Android библиотек приложением PDroid. Так, например, для доступа к телефонной связи класс TelephonyManager подменяется классом PrivacyTelephonyManager.

PrivacyTelephonyManager и аналогичные ему добавляемые классы подменяются PDroid-ом в классе ContextImpl. Однако в более актуальных версиях Android создание менеджеров было вынесено из ContextImpl в класс SystemServiceRegistry, который использует TelephonyFrameworkInitializer для инициализации TelephonyManager. Основываясь на перечисленном ранее, было решено для подмены контактов подменять инициализацию TelephonyManager в классе TelephonyFrameworkInitializer.

# 3. Реализация Magisk-модуля без использования Zygisk

## 3.1. Создание тестовых модулей Magisk

В рамках текущего проекта, опираясь на статью [26] и репозиторий [22], было создано четыре тестовых модуля.

### 3.1.1. Первый тестовый модуль

Первый модуль был создан на основе уже существующего модуля Magisk 1Controller Magisk Module [5] для проверки установки работающего программного обеспечения. В данном тестовом модуле были изменены детали описания модификации в файле `system.prop`.

### 3.1.2. Второй тестовый модуль

Второй тестовый модуль был шаблонным и никак не модифицировал систему. Все используемые в нем файлы писались с нуля, не основываясь на других рабочих модулях. На рисунке 1 изображен успешно установленный тестовый модуль.

### 3.1.3. Третий тестовый модуль

Третий тестовый модуль представлял собой модификацию системы, позволяющую обнулить список контактов на устройстве. Были реализованы несколько вариаций данного модуля, которые приведены ниже:

- переименование базы данных;
- подмена базы данных контактов на фиктивную пустую;
- копирование оригинальной базы данных для конкретного устройства и ее очистка.

Поскольку нужно именно заменить базу с контактами, от первого варианта было решено отказаться. Однако поскольку базы данных

на разных устройствах могут отличаться, было решено отказаться от второй реализации в пользу третьей. Вся логика клонирования и очистки базы контактов находилась в файле `customize.sh`, код был написан на языке программирования Shell.

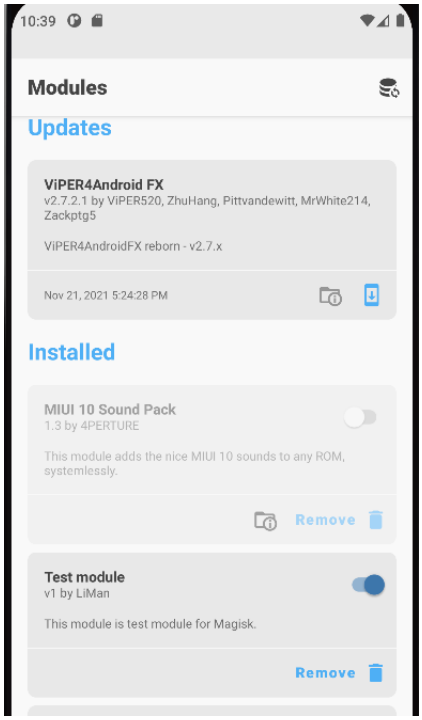


Рис. 1: Установленный тестовый модуль Magisk

### 3.1.4. Четвертый тестовый модуль

Так как в рамках текущего проекта база с контактами должна была подменяться выборочно для разных приложений, было решено модифицировать стандартные библиотеки Android. Для этого было необходимо научиться подменять файлы на устройстве. Для этой цели был реализован четвертый тестовый модуль, с помощью которого было проведено несколько успешных экспериментов по созданию новых и замене старых файлов на устройстве. Для этого использовалась директория `/system` в файловой структуре Magisk-модуля.

## 3.2. Поиск файла сборки измененного класса

В рамках данного проекта для выборочной подмены контактов в приложениях был написан класс PrivacyTelephonyManager, основываясь на похожем классе в PDroid. Для подмены вызова стандартного класса TelephonyManager на вызов нового класса необходимо было модифицировать исходный код TelephonyFromeworkInitializer. Но для того, чтобы произвести модификацию исходного класса в системе устройства, нужно найти, в каком файле он окажется после сборки, и понять как правильно его модифицировать.

### 3.2.1. Применение патча PDroid

Для того чтобы удостовериться в работоспособности патча, разобраться, как он работал, и найти файлы библиотек, на сборку которых повлиял патч, было решено применить патч из PDroid к операционной системе Android 4.1.1 (x86). Был выбран 4 релиз, поскольку автор, обновляющий PDroid для работы на последней на тот момент версии Android, использовал Android 4.1.1_4r [24].

Для сборки Android 4 была выполнена загрузка исходного кода с помощью специализированного инструмента `repo` [2]. Попытка запустить собранную систему с ядром, которое загружалось с используемой версией системы Android, окончилась зависанием на экране загрузки (stack on logo). Поэтому было решено поменять ядро. Согласно рекомендациям от Intel [23] было скачано ядро Linux 2.6 для запуска Android (x86). Android предназначен для устройств с сенсорным экраном, и по умолчанию он не поддерживает указатель мыши и, возможно, не поддерживает проводное соединение, поскольку большинство устройств Android используют беспроводную связь. Поэтому, чтобы создать Android для обычной платформы разработки приложений, было решено собрать ядро с поддержкой мыши и некоторых других функций.

После сборки ядра была применена сборка системы. Она проходила в два этапа. На первом этапе проходила сборка исходного кода, результатом которой являлся набор `img`-файлов таких, как `system.img`,

userdata.img и так далее. На втором этапе происходило создание пары vdi-образов на основе результатов первого этапа. Один для раздела с системой, другой для раздела, который монтируется как /data.

Далее ко всей директории исходного кода был применен патч и проведена аналогичная сборка. Тем самым, в результате получилось две пары виртуальных жестких дисков с патчем и без него. Было также замечено, что в системе с патчем появился файл framework2.jar. Анализ патча показал, что это связано с ограничением на количество методов в одном dex-файле, входящем во framework.jar, в который собирается framework/base. Часть поддиректорий, лежащих в framework, называемых в патче SECONDARY_FRAMEWORKS_SUBDIRS, были исключены из framework.jar и собраны отдельно в framework2.jar.

После патча были изменены core.jar, services.jar и framework.jar и добавлен framework2.jar.

### 3.2.2. Magisk модуль, использующий изменение стандартных библиотек

В рамках данной работы также был найден и проанализирован Magisk-модуль Bluetooth Library Patcher [20] для Samsung устройств, изменяющий стандартную библиотеку Bluetooth. Было выяснено, что он делает это путем изменения бинарного кода в bluetooth.default.so во время установки. Однако выяснить, как именно был получен необходимый кусок кода и место замены, не удалось.

### 3.2.3. Выводы

Основываясь на полученных результатах, было решено искать набор jar-файлов, который будет подменен. Было выявлено, что патч системы, модифицирующий отдельные классы стандартной библиотеки, может быть сведен к патчу набора jar-файлов, точный список которых необходимо определить, после чего декомпилировать их, применить патч и собрать для дальнейшей подмены с помощью модуля Magisk.

## 3.3. Изменение стандартных Android библиотек

В рамках данной работы для изменения стандартных библиотек сначала рассматривалось решение с помощью декомпиляции `jar`-файлов. Однако возникли следующие проблемы:

- поскольку декомпилированные классы использовали классы других библиотек, пришлось декомпилировать и другие библиотеки, которых было несколько;
- в декомпилированных файлах не было `make`-файлов, а также не было найдено алгоритма для сборки от компании Google;
- рассинхронизация собранной библиотеки и `odex`-файла, из-за чего устройство зависало на этапе загрузки.

В связи с этим было решено скачать, изменять и собирать исходные библиотеки Android. В рамках дипломного проекта была произведена поэтапная модификация стандартных классов. В качестве изменяемого класса было решено использовать `TelephonyManager.java`, поскольку работа с получением контактов происходит через него.

В качестве первого этапа модификации был изменен стандартный метод `getLine1Number()`. Он возвращал заведомо подготовленное значение 88005553535, игнорируя тело метода.

При попытке создать новую библиотеку, которая будет использоваться для подмены в стандартных классах, возникала ошибка при сборке. Для того чтобы убедиться в корректности работы созданного класса, было решено сделать промежуточный этап и поместить этот класс в `TelephonyManager.java`.

Убедившись, что проблема была не в коде, для корректной работы новой библиотеки были изменены файлы сборки системы: были добавлены названия новых классов, названия методов и полей.

## 3.4. Создание итогового модуля

Изначально было решено подменять модулем только файл `framework.jar`, однако при перезагрузке устройства, система зависла на экране загрузки. Изучив данную проблему, была найдена причина, которая заключалась в рассинхронизации `odex` и `jar`-файла. Для решения данной проблемы были добавлены все файлы в директории `system`, которые изменились после сборки системы.

Для проверки работы подмены библиотеки с помощью модуля было написано приложение, которое выводило только значение стандартного метода `getLine1Number()` класса `TelephonyManager`. На рисунке 2 и рисунке 3 приведены примеры работы этого приложения с выключенным модулем и включенным модулем соответственно.

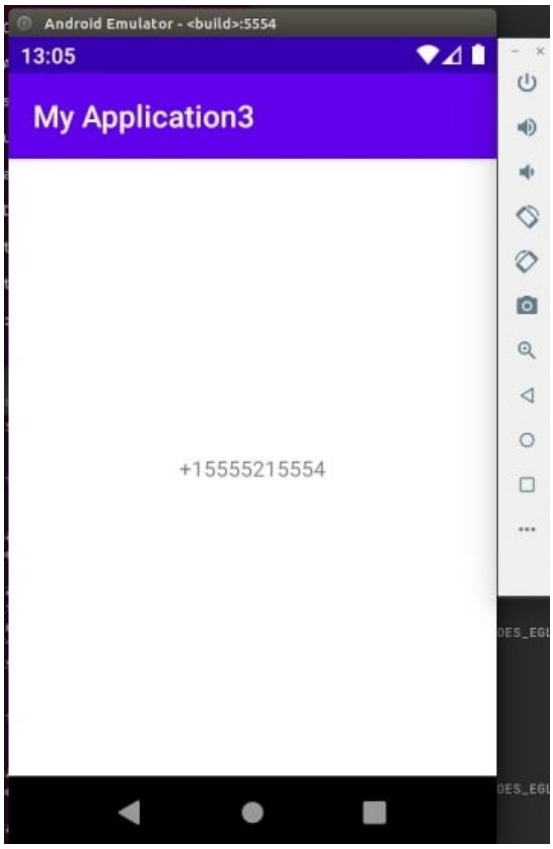


Рис. 2: Приложение с выключенным Magisk-модулем

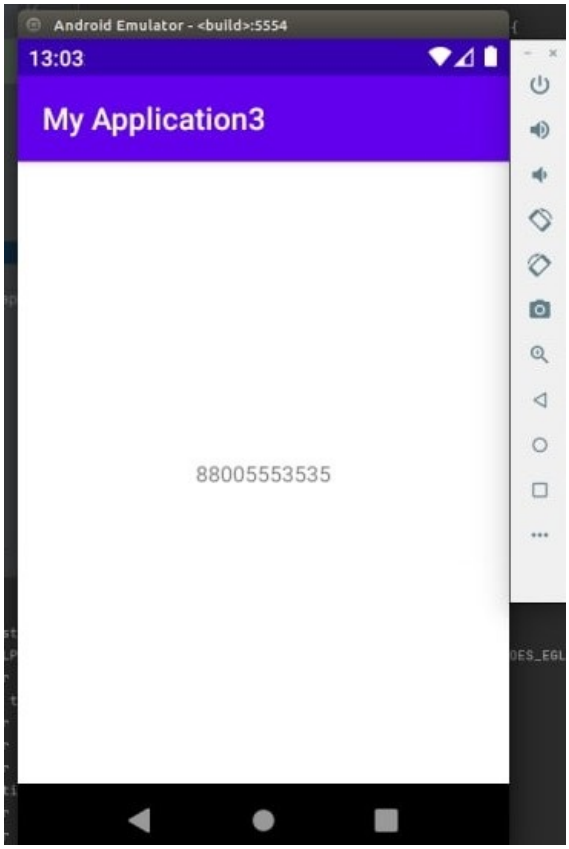


Рис. 3: Приложение с включенным Magisk-модулем

## 3.5. Приложение для выборочной подмены

В рамках данной работы было разработано приложение, устанавливающееся вместе с модулем. Оно предоставляет интерфейс для работы с установленными приложениями на устройстве. На данный момент это список приложений и флаг для каждого из них для включения или отключения подмены данных. На рисунке 4 продемонстрирован интерфейс данного приложения.

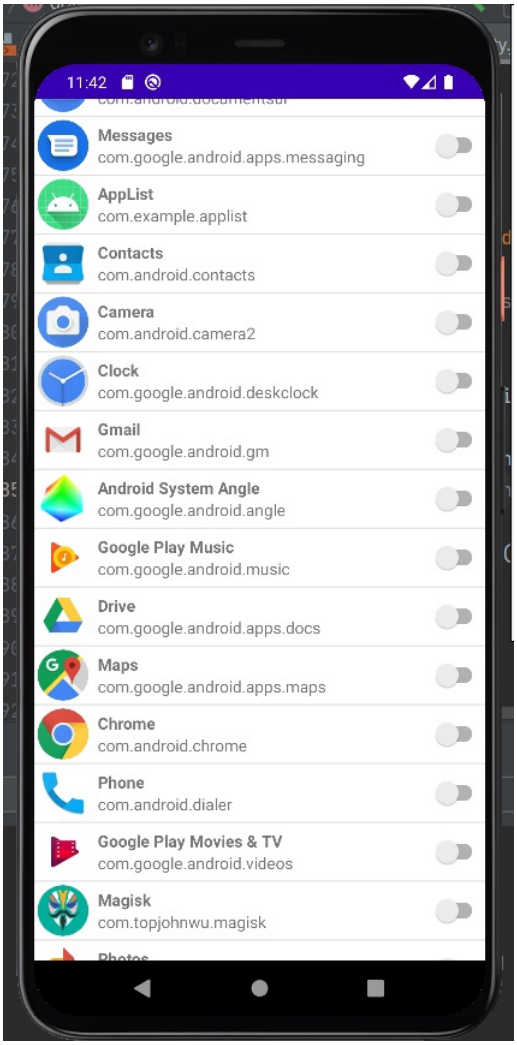


Рис. 4: Приложение для выборочной подмены

Для хранения значения флага каждого приложения была использована субд SQLite, диаграмма которой изображена на рисунке 5.

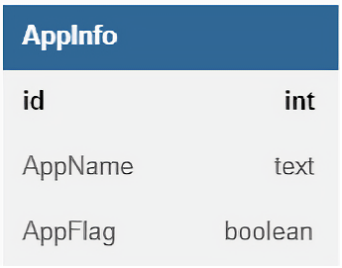


Рис. 5: Диаграмма таблицы с информацией о приложениях

## 3.6. Структура созданного модуля

В рамках текущего проекта был создан модуль со следующей структурой:

```
/data/adb/modules
├── $MODID
│   ├── module.prop
│   ├── system
│   │   ├── framework
│   │   └── app
└── └── customize.sh
```

В директории `/system/framework` содержатся файлы собранной модифицированной библиотеки `framework` для подмены на устройстве. А в директории `/system/app` Android-приложение в виде `apk`-файла. На рисунке 6 приведен установленный итоговый модуль.

## 3.7. Выводы

Несмотря на достигнутые результаты, данный модуль приводит к большим затратам временных ресурсов и памяти из-за установки и сборки исходного кода операционной системы Android. Для решения данной проблемы было решено использовать более современный метод, который использует Zygisk-модуль.

# 4. Реализация Zygisk-модуля

Из-за того, что для Magisk-модуля, не использовавшего Zygisk, использовалось Magisk-приложение 23 версии, было решено обновить его до последней на тот момент 25 версии, поскольку для использования Zygisk необходима версия приложения не ниже 24.

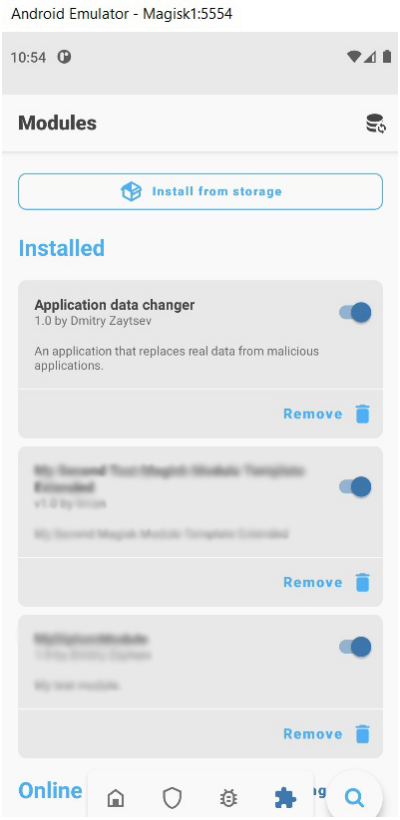


Рис. 6: Установленный итоговый Magisk-модуль

В рамках дипломного проекта, опираясь на официальный репозиторий [21] с примером и документацией, было создано несколько модулей.

## 4.1. Достоинства и недостатки Zygisk-модуля

Основной особенностью Zygisk-модуля является то, что он способен подменять нативные методы, что сильно упрощает работу для разработчика, поскольку нет необходимости патчить исходные библиотеки. Данное решение позволяет сократить зависимость от конкретной версии операционной системы Android, однако оно будет зависеть от версии приложения Magisk.

## 4.2. Сборка Zygisk-модуля

Zygisk-модуль для подмены поведения методов стандартных библиотек использует специальный Zygisk API, позволяющий подменять нативные методы, написанные на языке программирования C++. Для этого используется метод `hookJniNativeMethods()`. Также в Zygisk-модуле используется JNI [1]. Это собственный интерфейс Java, который определяет способ взаимодействия байт-кода, который Android компилирует из управляемого кода (написанного на языках программирования Java или Kotlin), с машинным кодом (написанным на C/C++).

Для сборки Zygisk-модуля необходимо использовать NDK 21 версии и старше. Результатом такой сборки будут являться четыре `.so` файла, являющимися собранными библиотеки, которые необходимо положить в директорию `/zygisk` Magisk-модуля.

## 4.3. Модуль, подменяющий контакты

В рамках дипломного проекта был реализован Zygisk-модуль, позволяющий выборочно для приложений подменять контакты.

Во время разработки были найдены нативные JNI методы, которые используются для получения в том числе данных о контактах пользователя. В данном примере было решено использовать метод `nativeGetString` класса `android/database/CursorWindow`, который получает значение из базы данных.

В качестве решения было сделано два варианта: с заменой изначального значения на `null` и на фиксированное значение. Примеры работы модуля приведены на рисунке 7.

## 4.4. Модуль, подменяющий сообщения

В рамках дипломного проекта также был реализован Zygisk-модуль, позволяющий выборочно для приложений подменять сообщения пользователя.

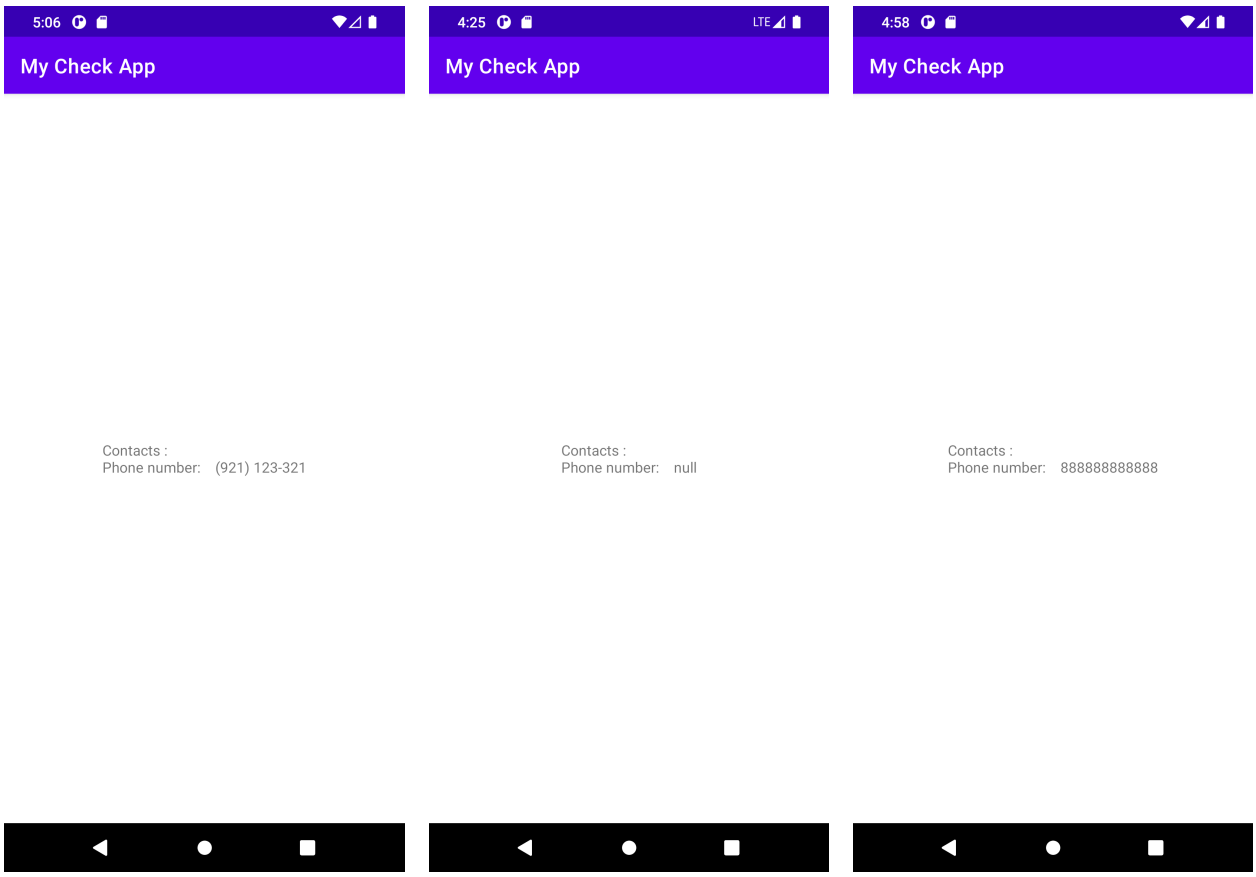


Рис. 7: Работа приложения получения контактов без модуля; с модулем, подменяющим значение на null; с модулем, подменяющим значение на фиксированное значение

В данном примере было также решено использовать метод `nativeGetString` класса `android/database/CursorWindow`.

Как и в предыдущем примере, в качестве решения было сделано два варианта: с заменой изначального значения на `null` и на фиксированное значение. Примеры работы модуля приведены на рисунке 8.

## 4.5. Псевдокод получившихся Zygisk-модулей

Для написания всех Zygisk-модулей использовался следующий алгоритм, приведенный в листинге 1.

Для подмены нативного метода `native_method` необходимо создать и реализовать свой метод `my_method`. После чего в классе, унаследованном от `zygisk::ModuleBase`, переопределить функцию `preAppSpecialize`, поскольку изменение нативных методов необходимо до вызова приложе-

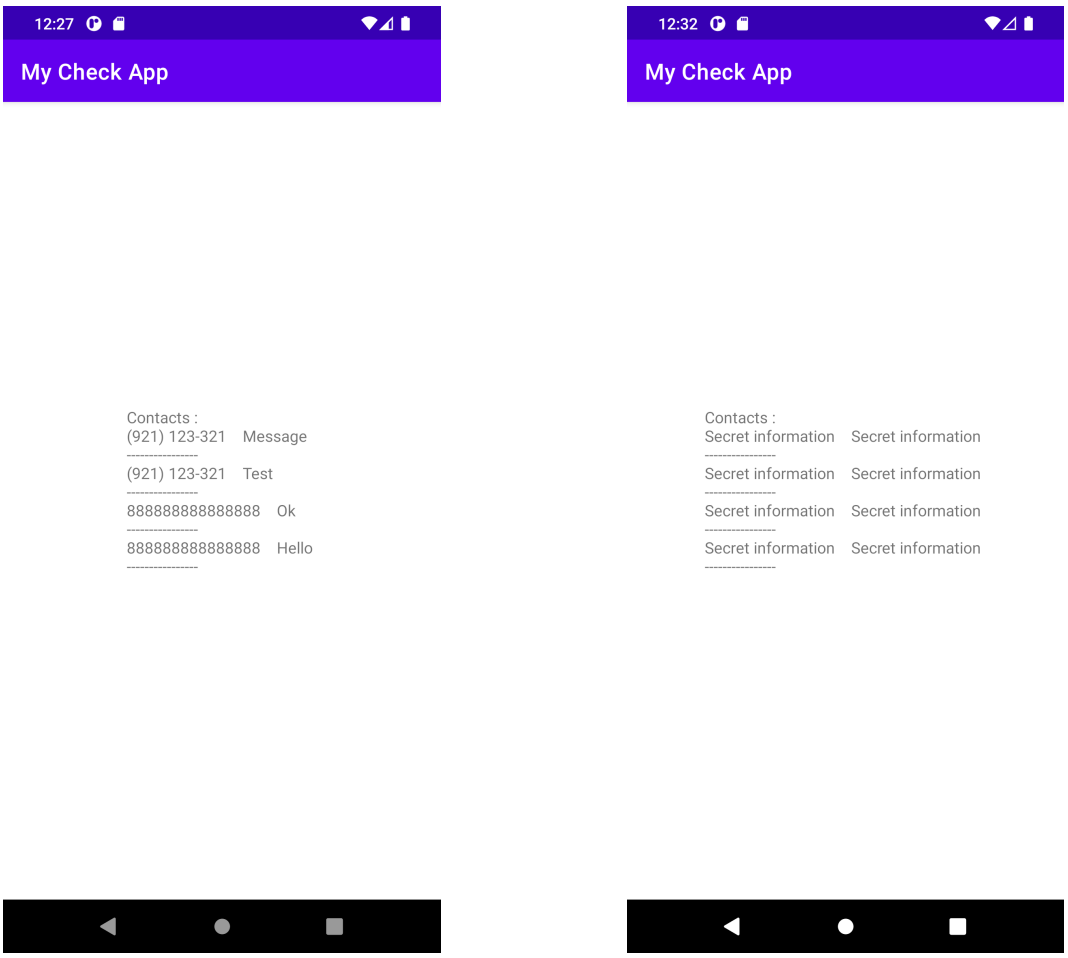


Рис. 8: Работа приложения получения сообщений без модуля; с модулем, подменяющим значение на фиксированное значение

ния. В теле этой функции для всех методов, которые запускаются из вредоносных приложений, делать подмену. Для этого необходимо составить список JNI-методов и выполнить метод `hookJniNativeMethods`.

```
static jType my_method(params) //Реализация собственнойфункции
{
    result = doSomething();
    return result;
};

class ExampleModule : public zygisk::ModuleBase {
    void preAppSpecialize(zygisk::AppSpecializeArgs *args) override
    {
        if (applicationName in [BadApplications])
        {
            JNINativeMethod methods[] = {
                {"native_method", "(input_signature)_output_signature",
                (void*)(my_method)}
```

```
            };

            api->hookJniNativeMethods(env, "android_path", methods, 1);
        }
    }
}
```

**Листинг 1:** Псевдокод Zygisk-модулей

## 4.6. Структура получившихся Zygisk-модулей

В рамках текущего проекта были созданы модули со следующей структурой:

```
/data/adb/modules
├── $MODID
│   ├── module.prop
│   ├── system
│   │   └── app
│   ├── zygisk
│   │   ├── arm64-v8a.so
│   │   ├── armeabi-v7a.so
│   │   ├── x86.so
│   │   └── x86_64.so
└── └── customize.sh
```

В директории `/zygisk` содержатся файлы собранных нативных библиотек для подмены на устройстве. А в директории `/system/app` Android-приложение в виде `apk`-файла. На рисунке 9 приведен установленный итоговый модуль.

## 4.7. Выводы

Достигнутые результаты показали, что с помощью Zygisk-модули разрабатывать и сопровождать менее затратно по времени и памяти чем использование патча и подмена собранных библиотек с помощью модуля Magisk, не использовавшего Zygisk. Однако отсутствие достаточного

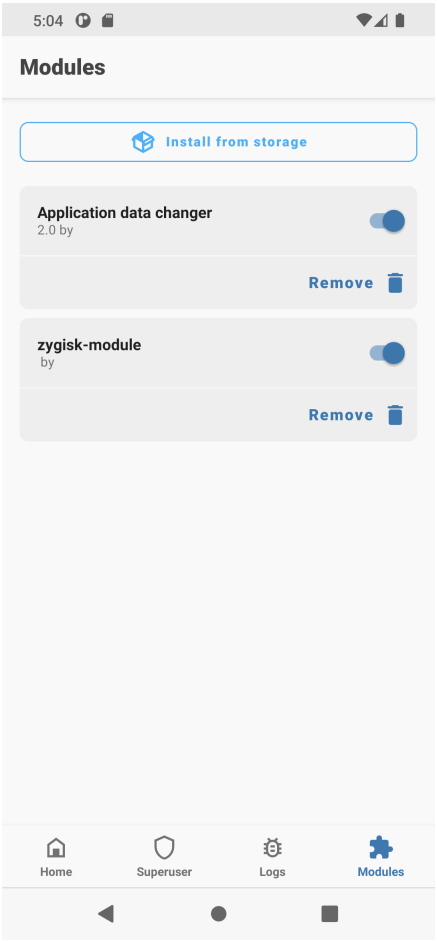


Рис. 9: Установленный итоговый Zygisk-модуль

числа примеров и поиск необходимых для замены нативных методов значительно затрудняют процесс разработки.

# 5. Сравнение подходов подмены пользовательских данных

Разработанные двумя методами модули показали перечисленные ниже достоинства и недостатки каждого решения.

## 5.1. Модуль Magisk, не использовавший Zygisk

Достоинства:

- поиск необходимого метода для замены не требует значительных временных затрат;
- не зависит от версии Magisk.

Недостатки:

- для реализации подмены требуются значительные затраты временных ресурсов и памяти, связанные с установкой и сборкой исходного кода Android;
- зависит от версии операционной системы Android.

## 5.2. Модуль с использованием Zygisk

Достоинства:

- не требует значительных затрат временных ресурсов и памяти при сборке, что значительно упрощает сопровождение модуля;
- не зависит от версии операционной системы Android;
- является современным решением.

Недостатки:

- поиск необходимого нативного метода;
- крайне ограниченное количество примеров;

- малоинформативная документация по разработке модуля;
- зависимость от версии Magisk.

## 5.3. Выводы

Поскольку разработка и поддержка модуля являются более важными критериями в связи с дальнейшим развитием данного модуля, перспективнее оказался метод, использующий Zygisk.

# Заключение

В рамках дипломной работы были достигнуты результаты, перечисленные ниже:

- подготовлена среда разработки и тестирования для Magisk;
- проведен анализ изменения библиотек ПО PDroid Privacy Protection:
  - выделены библиотеки, отвечающие на запросы о персональных данных средствами Android API;
- подготовлена библиотека для подмены контактов;
- локализована собранная библиотека:
  - проведен анализ процесса патча стандартных системных библиотек устройства;
- реализован работающий прототип, модифицирующий стандартные библиотеки Android, являющийся Magisk-модулем без использования Zygisk;
- реализован работающий прототип с использованием Zygisk;
- реализовано Android приложение, устанавливающееся с Magisk-модулем для выборочной работы модуля;
- проанализированы подходы решения с использованием Zygisk и без него.

Весь код модулей, приложения, а также патч-файл для изменения библиотеки находится в репозитории [4].

В будущем данный проект может быть развит путём расширения функционала, связанного с заменой пользовательских данных, а также улучшением приложения, отвечающего за выбор вредоносных приложений.

# Список литературы


[1] Android. JNI tips. — URL: https://developer.android.com/training/articles/perf-jni (дата обращения: 2023-05-03).

[2] Android. Repo. — URL: https://android.googlesource.com/tools/repo (дата обращения: 2022-05-31).

[3] Android. Официальный Android. — URL: https://www.android.com/ (дата обращения: 2021-12-26).

[4] Dima Zaicev. Репозиторий дипломного проекта. — URL: https://github.com/ZaicevDima/Magistracy-diploma/tree/Contacts-module (дата обращения: 2022-05-30).

[5] Electric1447. Репозиторий Magisk модуля 1Controller. — URL: https://github.com/Magisk-Modules-Repo/OneController (дата обращения: 2022-05-29).

[6] Forums XDA. Форум с описанием LBE Privacy Guard. — URL: https://forum.xda-developers.com/t/app-root-lbe-privacy-guard-most-powerful-privacy-protection-app-1091065/ (дата обращения: 2021-12-26).

[7] Forums XDA. Форум с описанием PDroidPrivacy Protection. — URL: https://forum.xda-developers.com/t/mod-vrblk3-pdroid-privacy-protection-keep-apps-from-stealing-yo-2043699/ (дата обращения: 2021-12-26).

[8] Google. Google Git репозиторий. — URL: https://android.googlesource.com/platform/libcore/ (дата обращения: 2021-12-26).

[9] LDPlayer. Официальный сайт LDPlayer. — URL: https://ru.ldplayer.net/ (дата обращения: 2021-12-26).

[10] Lab Adventures. Статистика мобильных приложений 2021: загрузки, тренды и доходность

индустрии. — URL: https://exlibris.ru/news/statistika-mobilnyh-prilozhenij-2021-zagruzki-trendy-i-dohodnost (дата обращения: 2021-12-26).

[11] LineageOS. Официальный LineageOS. — URL: https://www.lineageos.org (дата обращения: 2021-12-26).

[12] M66B. Репозиторий XPrivacyLua. — URL: https://github.com/M66B/XPrivacyLua (дата обращения: 2021-12-26).

[13] Magisk. Официальный сайт Magisk. — URL: https://magisk.me/ (дата обращения: 2021-12-26).

[14] Magisk. Руководство по разработке. — URL: https://topjohnwu.github.io/Magisk/guides.html (дата обращения: 2022-05-29).

[15] Oracle. Официальный сайт VirtualBox. — URL: https://www.virtualbox.org/ (дата обращения: 2021-12-26).

[16] Protection PDroid Privacy. Процесс изменения фреймворков. — URL: https://www.dropbox.com/s/hm5ouwf0sggo8de/PDroid-Source-2.3.4.zip?file_subpath=%2Fframeworks.patch (дата обращения: 2021-12-26).

[17] Repository Xposed Module. Официальный сайт Xposed Module Repository. — URL: https://github.com/Xposed-Modules-Repo (дата обращения: 2022-12-27).

[18] Shizuku. Официальный сайт Shizuku. — URL: https://shizuku.rikka.app/ (дата обращения: 2022-12-25).

[19] Studio Android. Официальный сайт Android Studio Emulator. — URL: https://developer.android.google.cn/studio/run/emulator?hl=en-GB (дата обращения: 2021-12-26).

[20] Trouillot Arthur. Репозиторий Magisc модуля Bluetooth Library Patcher. — URL: https://github.com/Magisk-Modules-Repo/BluetoothLibraryPatcher (дата обращения: 2022-05-31).

[21] Wu John. Репозиторий с описанием создания модуля Zygisk. — URL: https://github.com/topjohnwu/zygisk-module-sample (дата обращения: 2023-05-03).

[22] Zackptg5. Репозиторий с описанием создания модуля Magisk. — URL: https://github.com/Zackptg5/MMT-Extended (дата обращения: 2021-12-26).

[23] Zone Intel Developer. Hands-on Notes:Build Android-x86 ICS 4 Virtualbox from Google Virtualbox Target and Intel Kerneln. — URL: https://web.archive.org/web/20151114135757/https://software.intel.com/en-us/blogs/2012/03/06/hands-on-notesbuild-android-x86-ics-4-virtualbox-from-google-vi (дата обращения: 2012-03-06).

[24] gsbabil. Репозиторий с процессом изменения системных библиотек. — URL: https://github.com/gsbabil/PDroid-AOSP-JellyBean (дата обращения: 2022-05-31).

[25] topjohnwu. Репозиторий с релизами Magisk. — URL: https://github.com/topjohnwu/Magisk/releases (дата обращения: 2022-12-25).

[26] Подкопаев Дмитрий. Magisk. Модифицируем прошивку Android с комфортом // Хакер. — 2017. — . — URL: https://xakep.ru/2017/05/25/magisk-android-firmware-customization/ (дата обращения: 2021-12-26).

[27] Чебышев Виктор. Развитие информационных угроз в первом квартале 2021 года. Мобильная статистика // SecureList. — 2021. — . — URL: https://securelist.ru/it-threat-evolution-q1-2021-mobile-statistics/101595/ (дата обращения: 2021-12-26).