

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Обеспечение информационной безопасности мобильных устройств»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

«Создание приложения и поиск URL, конечных точек и секретов в APK-файлах (Kotlin)»

**Выполнил:**

Чу Ван Доан, студент группы N3347



\_\_\_\_\_  
(подпись)

**Проверил:**

Федоров Иван Романович

\_\_\_\_\_  
(отметка о выполнении)

\_\_\_\_\_  
(подпись)

Санкт-Петербург

2025 г.

## СОДЕРЖАНИЕ

Содержание.....	2
-----------------	---

## ВВЕДЕНИЕ

С ростом популярности мобильных устройств и приложений возрастает важность понимания как процесса разработки, так и анализа безопасности мобильных программ. Особенно актуальной становится задача изучения структуры APK-файлов, а также методов поиска чувствительных данных, таких как URL-адреса, API-ключи и конечные точки взаимодействия с внешними сервисами.

Целью данной лабораторной работы является получение практических навыков в области разработки мобильных приложений на языке Kotlin, а также базового анализа безопасности путём декомпиляции и исследования APK-файлов. Выполнение работы предполагает создание простого Android-приложения, включающего экран с персональными данными, экран, работающий с REST API, и экран с формой обратной связи.

В рамках лабораторной работы рассматриваются инструменты анализа APK-файлов, такие как ApkTool и apkLeaks, а также онлайн-сервис VirusTotal. Сравнение декомпилированных версий приложений с и без обфускации позволяет оценить эффективность методов сокрытия логики кода. Кроме того, особое внимание уделяется безопасности мобильных приложений: поиск открытых ключей и конфиденциальной информации в декомпилированном коде позволяет лучше понять угрозы, связанные с недостаточной защитой.

Таким образом, выполнение данной работы способствует углублённому пониманию жизненного цикла Android-приложений, повышает уровень осведомлённости в области мобильной безопасности и закладывает основы для последующего изучения реверс-инжиниринга и анализа вредоносного ПО.

### Задание

1. В Android Studio создать проект мобильного приложения на Kotlin.
2. Разработанное приложение должно содержать следующие экраны:
  - Экран с выводом ФИО студента
  - Экран, отображающий данные, полученные через вызов к REST API (можно использовать любой доступный публичный REST API)
  - Экран с формой обратной связи (можно реализовать взаимодействие с базой данных)
3. Получить apk. файл двумя способами: с обфускацией и без обфускации
4. Декомпилировать полученные apk с помощью ArkTool, сравнить результаты
5. Просканировать разработанное приложение утилитой apkLeaks
6. Просканировать любое стороннее приложение (.apk) утилитой apkLeaks и сравнить результат
7. Проверить разработанное в рамках ЛР приложение через VirusTotal

## ХОД РАБОТЫ

### 1. Создание проекта в Android Studio и получение .apk файла

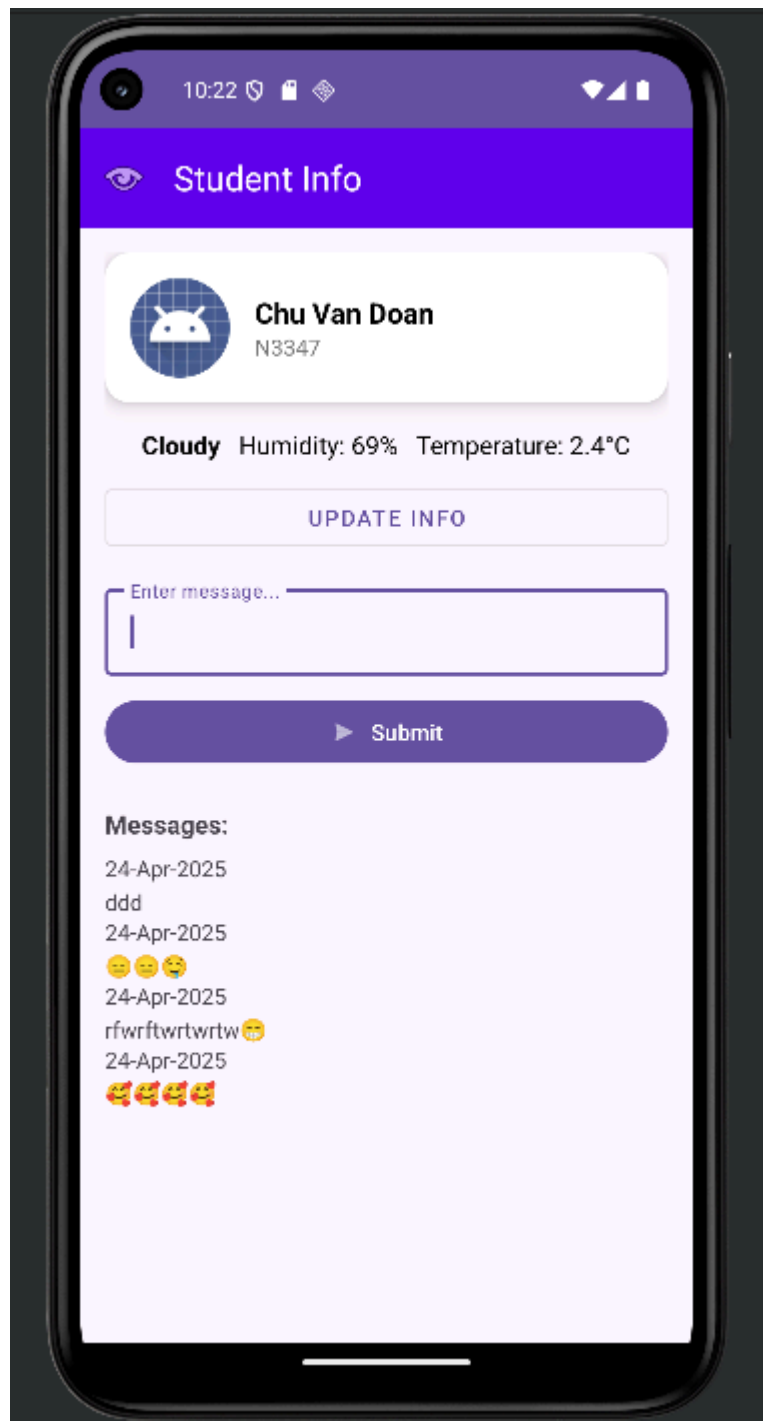


Рисунок 1 - Простенькое приложение

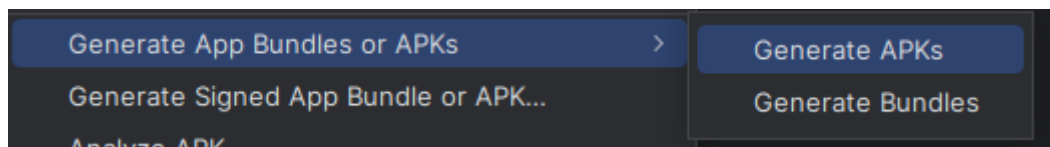


Рисунок 2 - Получение арк. файл с обфускацией без обфускации

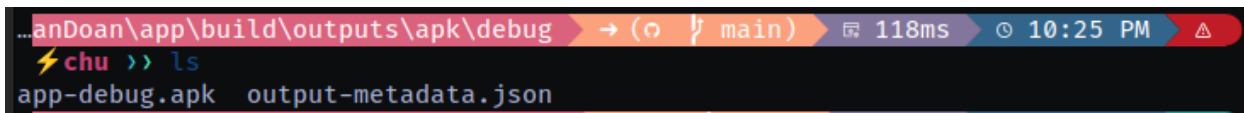


Рисунок 3 - арк. файл с обфускацией без обфускации

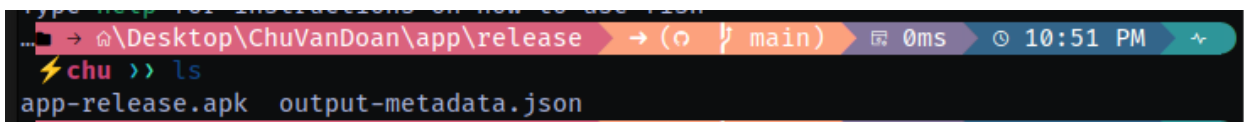


Рисунок 4 - арк. файл с обфускацией обфускации

## 2. Декомпилировать приложение с помощью APKTool

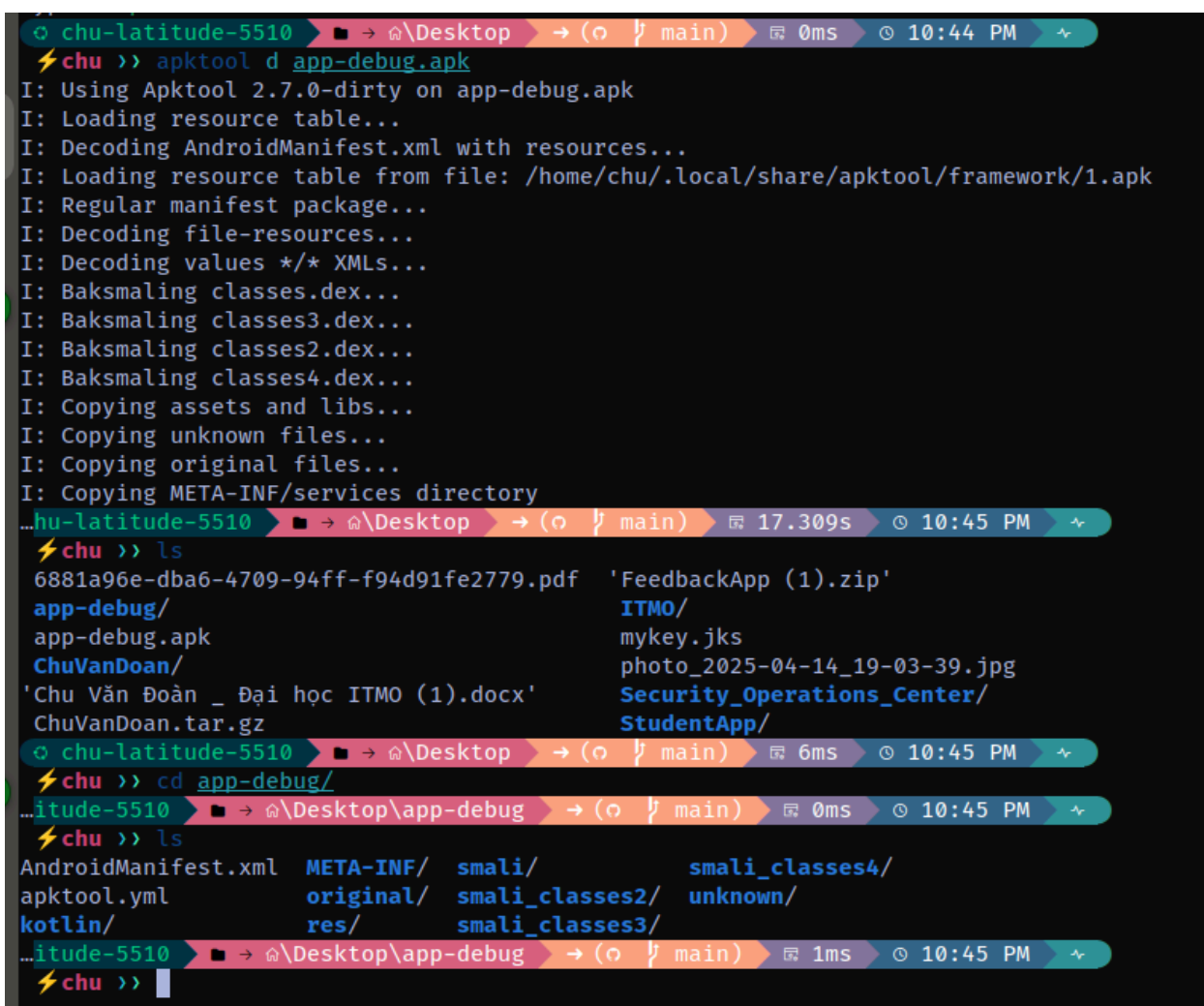


Рисунок 5 - Декомпиляция приложения

### 3. Сканирование приложения утилитой APKLeaks

```
?chu >> ~/.local/bin/apkleaks -h
```

```
      _/_/\_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_\n     /-_-|\_|_)| |'/_/_/_/_/_/_/_/_/_/_/_\n    /--_\_\\_|_/|.\\_|_|_|_|_|_|_|_|_|_|_|_\n   /_/\\_\\_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_\nv2.6.3\n--\nScanning APK file for URIs, endpoints & secrets  
(c) 2020-2024, dwisiswant0
```

```
usage: apkleaks [-h] -f FILE [-o OUTPUT] [-p PATTERN] [-a ARGS] [--json]
```

```
options:
```

```
-h, --help          show this help message and exit
```

```
-f FILE, --file FILE  APK file to scanning
```

```
-o OUTPUT, --output OUTPUT
```

```
                    Write to file results (random if not set)
```

```
-p PATTERN, --pattern PATTERN
```

```
                    Path to custom patterns JSON
```

```
-a ARGS, --args ARGS  Disassembler arguments (e.g. --threads-count 5 --deobf)
```

```
--json              Save as JSON format
```

Рисунок 6 - Установка утилиты






```
⚡chu >> cat /tmp/apkleaks-qhstxgk1.txt
[IP_Address]
- 127.0.0.1

[JSON_Web-Token]
- androidGradlePluginVersion=8.2.2

[LinkFinder]
- /...
- /NULL
- /NULL
- /data/misc/profiles/cur/0
- /data/misc/profiles/cur/0/
- /data/misc/profiles/ref/
- /index.html
- /proc/self/fd/
- M/d/yy
- META-INF/services/
- activity_choser_model_history.xml
- current.json
- dexopt/baseline.prof
- http://api.weatherapi.com/v1/
- http://schemas.android.com/apk/res-auto
- http://schemas.android.com/apk/res/android
- share_history.xml
```

Рисунок 8 - Результаты сканирования

#### 4. Просканировать любое стороннее приложение утилитой



```
chu >> cat /tmp/apkleaks- 8a90lrh.txt
[Authorization_Basic]
- basic =
- basic events
- basic library
- basic script
- basic serif

[Facebook_Secret_Key]
- FBF_HASH = "2438bce1ddb7bd026d5ff89f598b3b5e
- FBI_HASH = "a4b7452e2ed8f5f191058ca7bbfd26b0
- FBL_HASH = "df6b721c8b4d3b6eb44c861d4415007e
- FBR2_HASH = "cc2751449a350f668590264ed7669269
- FBR_HASH = "8a3c4b262d721acd49a4bf97d5213199

[Firebase]
- symbolab-calculator.firebaseio.com

[Generic_API_Key]
- ApiKey" android:value="7822a5f3a0fcfea4c32d3af859dbf62cfb835fde"

[Google_API_Key]
- AIzaSyBGH4V9ctA_bGhKpph9-uBu7LTt-kCoLUA

[Google_Cloud_Platform_OAuth]
- 495389923697-sfmpvs280bujr4koiskulf245vnacrl.apps.googleusercontent.com

[IP_Address]
- 0.0.0.0
- 10.0.2.2
- 10.100.102.3
- 104.200.18.2
- 104.237.129.1
- 104.237.129.7
- 127.0.0.1
- 143.42.178.1
- 170.187.161.1
- 173.255.235.1
- 192.168.1.1
- 192.168.68.1
- 198.58.98.1
- 45.33.28.2
- 69.164.192.7

[JSON_Web_Token]
```

Рисунок 9 - Результаты сканирования file symbolab apk

Результаты анализа APK-файла приложения Symbolab v10.9.0 с помощью утилиты APKLeaks показали наличие множества потенциальных утечек конфиденциальной информации. В частности, были обнаружены строки с API-ключами, OAuth client ID, адреса Firebase, а также как внутренние, так и внешние IP-адреса. Кроме того, было

выявлено множество URL и конечных точек API, включая обработку пользовательских данных, платежей и заметок. Также найдены хэши, связанные с Facebook SDK и Google API, которые могут быть использованы злоумышленниками при отсутствии надлежащей защиты. Это свидетельствует о недостаточной обфускации приложения и слабой защите внутренней логики, что увеличивает риск статического анализа и несанкционированного доступа к API. Рекомендуется внедрение дополнительных мер безопасности: шифрование, обфускация и строгий контроль доступа, чтобы минимизировать угрозу возможной эксплуатации.

5. Проверка разработанного приложения с помощью VirusTotal

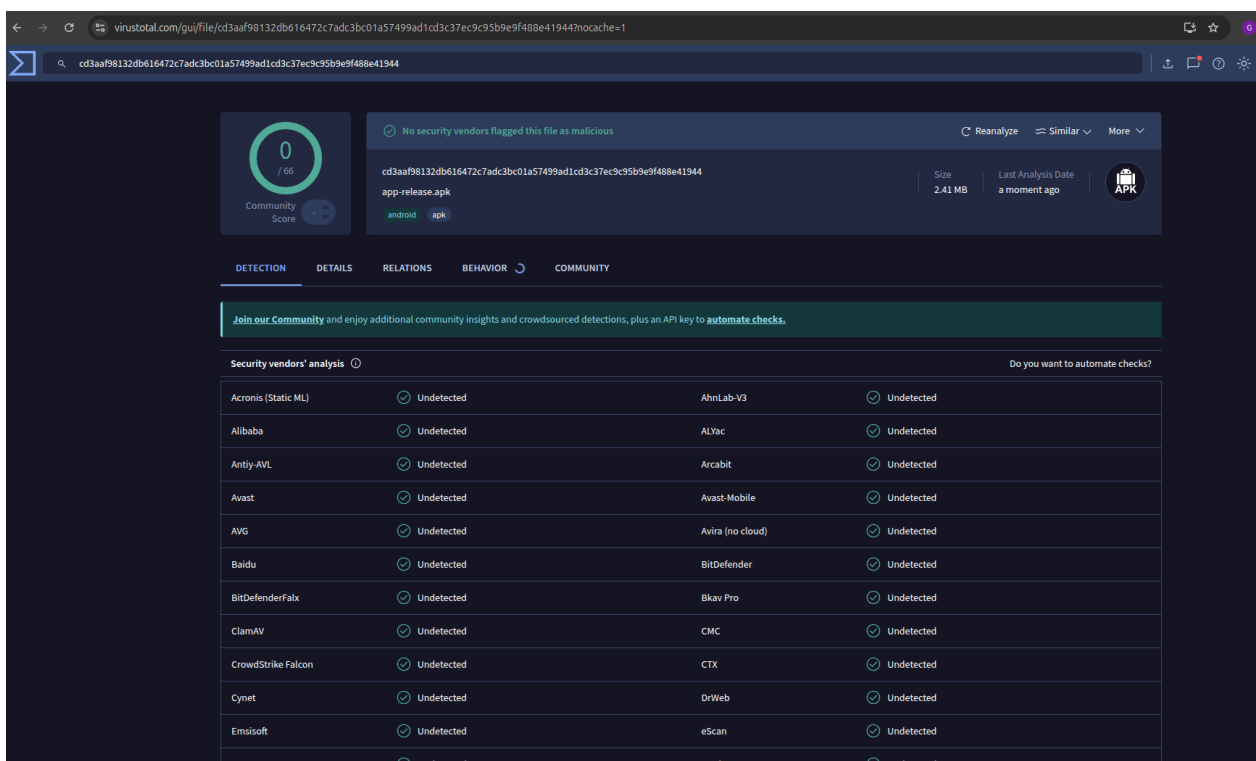


Рисунок 10 - Результаты сканирования с помощью VirusTotal

После проверки на VirusTotal, угроз в безопасном приложении не было найдено. В результате, 66/66 антивирусы сказали, что наше приложение безопасно.

## **ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы было разработано Android-приложение, реализующее задачи вывода ФИО студента, отображения данных через REST API и предоставления формы обратной связи. Для проверки безопасности приложения использовались инструменты, такие как ApkTool, apkleaks и VirusTotal. Анализ результатов позволил выявить особенности приложения и провести необходимые улучшения.

### **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. <https://github.com/ashishb/android-malware>
2. <https://xakep.ru/2014/05/21/excuse-in-android-architecture/>
3. <https://www.virustotal.com/gui/home/upload>