МИНОБРНАУКИ РОССИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГУ»)
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ, ИНФОРМАТИКИ И МЕХАНИКИ

Дисциплина «Безопасность мобильных устройств»
Направление 02.04.02 «Фундаментальная информатика и информационные технологии»
Лабораторная работа №3

Выполнила магистры 2 курса 13 группы:
Бутузова Д. О.
Преподаватель: Вернер Е. С.

Воронеж 2025

Введение

В данном отчёте рассматривается реализация безопасного TLS-соединения в Android-приложении на Kotlin.

1. Цели

Обеспечение безопасного обмена данными между мобильным приложением и сервером через протокол TLS (Transport Layer Security) с использованием библиотеки OkHttp. Демонстрация корректной обработки HTTPS-запросов, парсинга JSON и отображения данных в UI.

2. Алгоритм TLS

TLS — это криптографический протокол, обеспечивающий безопасную передачу данных в интернете. Он используется для защиты соединений между клиентом (например, вашим приложением) и сервером.

Основные функции TLS:

- Шифрование данных — предотвращает перехват информации.
- Аутентификация сервера (и клиента) — проверка подлинности сертификатов.
- Целостность данных — защита от подмены пакетов.

TLS 1.2/1.3 включены по умолчанию в современных Android-приложениях.

При установке HTTPS-соединения Android автоматически проверяет подлинность SSL-сертификата сервера с помощью системы доверия, встроенной в ОС. Это происходит следующим образом:

- Встроенный в Android TrustManager сравнивает цепочку сертификатов сервера с доверенными корневыми сертификатами, хранящимися в системном хранилище.
- Если сертификат сервера подписан одним из доверенных корневых удостоверяющих центров (CA), соединение считается безопасным.
- Если сертификат самоподписан, просрочен, подделан или недействителен, соединение прерывается, и вызывается onFailure().

Таким образом, OkHttpClient использует реализацию X509TrustManager по умолчанию, обеспечивая защиту от атак типа "человек посередине" (MITM).

3. Разработка программы на языке Kotlin

3.1 Настройка TLS-соединения:

Используется стандартный клиент OkHttp (OkHttpClient()), который автоматически поддерживает TLS 1.2/1.3 и проверку сертификатов сервера.

HTTPS-запрос к защищенному ресурсу https://reqres.in/api/users/2 гарантирует шифрование данных.

OkHttp проверяет сертификаты сервера, что исключает атаки типа MITM (Man-in-the-Middle).

В AndroidManifest.xml добавлено разрешение для выхода в интернет:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Функция обработки безопасного запроса в MainActivity:

```
 private fun sendSecureRequest() {
        val request = Request.Builder()
            .url("https://reqres.in/api/users/2")
            .build()

        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                Log.e("TLS", "Ошибка соединения: ${e.message}")
            }

            override fun onResponse(call: Call, response: Response) {
                response.use {
                    if (!it.isSuccessful) {
                        Log.e("TLS", "Ошибка: ${it.code}")
                    } else {
                        val json = JSONObject(it.body?.string() ?: "")
                        val data = json.getJSONObject("data")

                        val firstName = data.getString("first_name")
                        val lastName = data.getString("last_name")
                        val email = data.getString("email")
                        val avatar = data.getString("avatar")

                        runOnUiThread {
                            findViewById<TextView>(R.id.nameTextView).text =
"$firstName $lastName"
                            findViewById<TextView>(R.id.emailTextView).text =
email
                            Glide.with(this@MainActivity)
                                .load(avatar)
                                .circleCrop()

.into(findViewById<ImageView>(R.id.avatarImageView))
                        }
                    }
                }
            }
        })
    }
```

3.2 Асинхронный запрос

Для избежания блокировки UI применяется асинхронный вызов enqueue().
Обработка ошибок:

- onFailure: Логирование проблем сети (например, отсутствие интернета).
- onResponse: Проверка кода ответа (isSuccessful).

3.3 Парсинг и отображение данных

Ответ сервера парсится с помощью JSONObject.
Данные пользователя (имя, email, аватар) выводятся в TextView и ImageView.
Библиотека Glide загружает изображение, применяет круглое обрезание (circleCrop()).

3.4 Вёрстка интерфейса

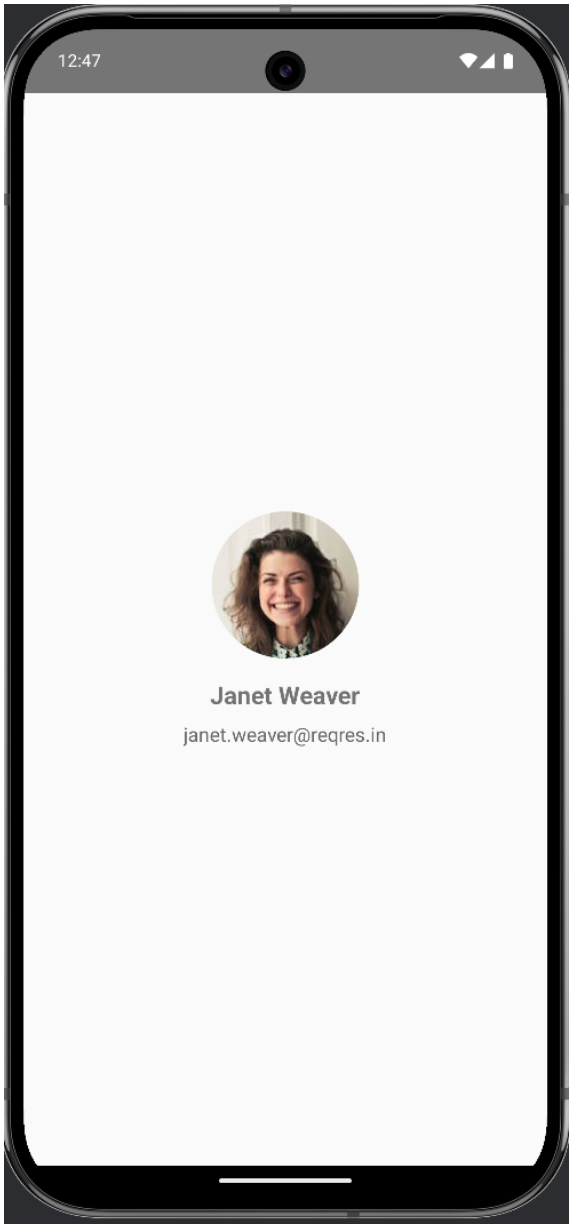
В xml-файле использованы следующие элементы:

- LinearLayout с центрированными элементами.
- ImageView для аватара (120x120 dp).
- Два TextView для имени и email.

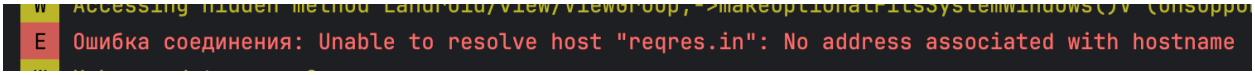
4. Пример работы приложения

При запуске приложения отправляется GET-запрос на https://reqres.in/api/users/2.

При успешном ответе имя и фамилия выводятся в nameTextView, email отображается в emailTextView, аватар загружается через Glide и отображается на экране.



В случае возникновения ошибки в логи выводятся код ошибки или описание проблемы, например при отсутствии интернет-соединения:



Вывод

В ходе выполнения лабораторной работы была успешно реализована безопасная передача данных между Android-приложением и сервером с использованием TLS (Transport Layer Security). Приложение безопасно обменивается данными с сервером, а код соответствует современным стандартам защиты информации.