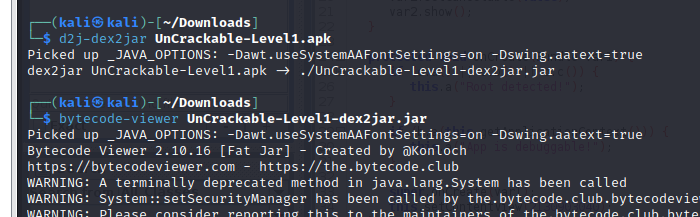
ЗВОРОТНА РОЗРОБКА ТА АНАЛIЗ ШКIДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

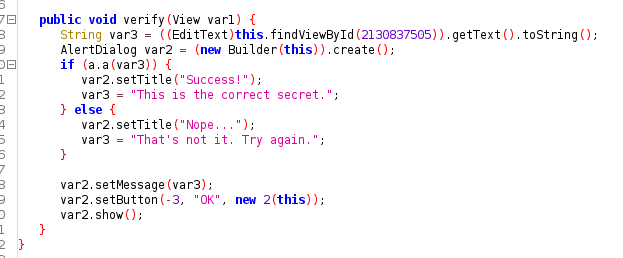
Лабораторна робота №8

**Проаналiзуйте UnCrackable Mobile Apps [181], Hacking Playground [182] з OWASP MSTG [183].**

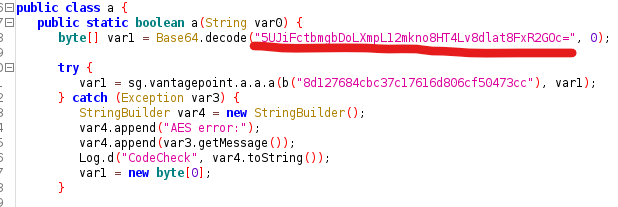
**Level 1**

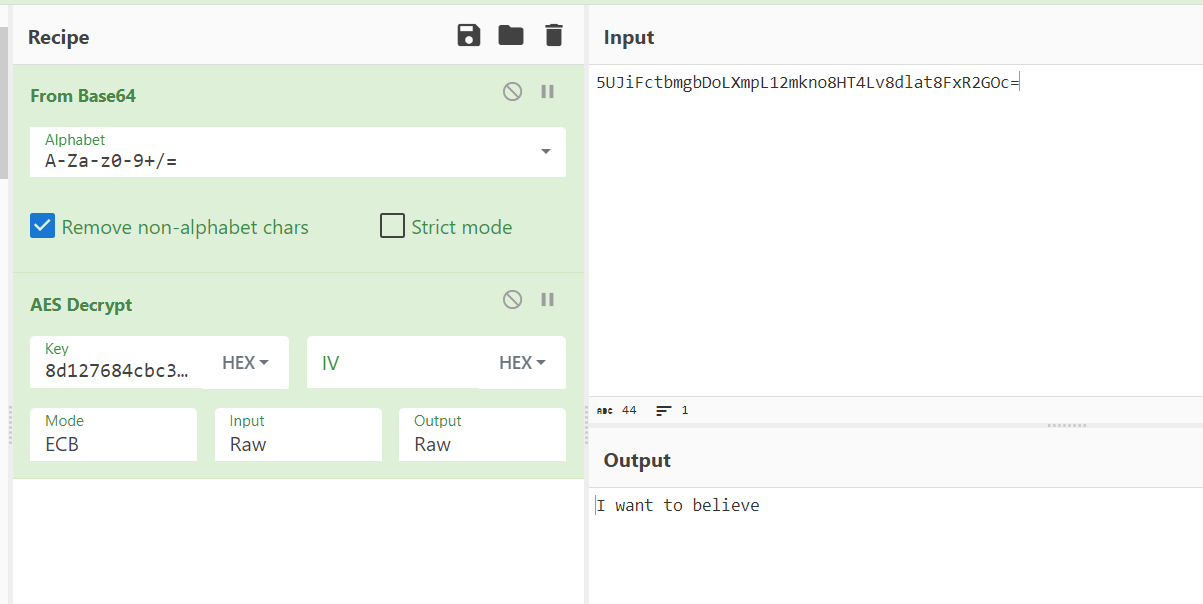
****

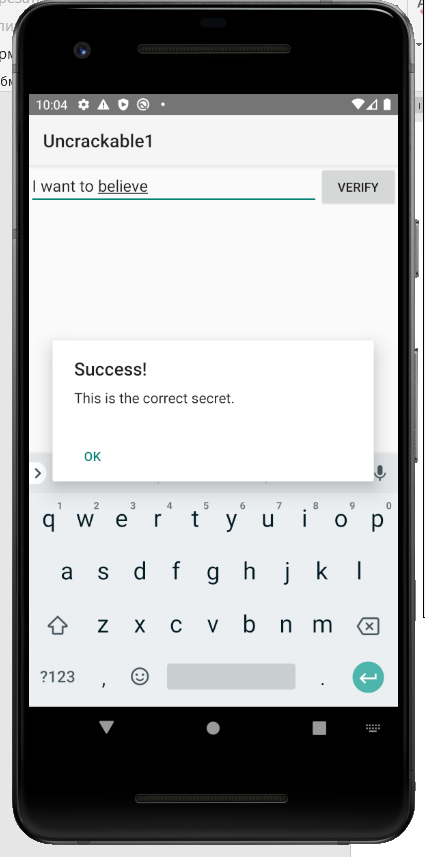
Бачимо функцію verify, яка перевіряє введений текст



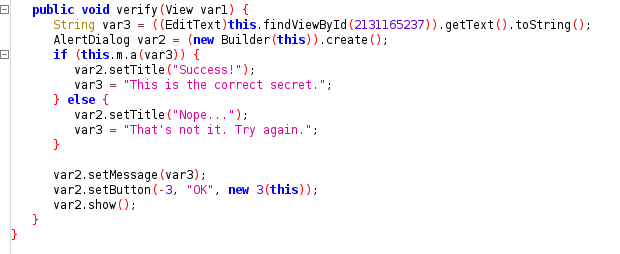
Спробуємо декодувати цю строку







**Level 2**

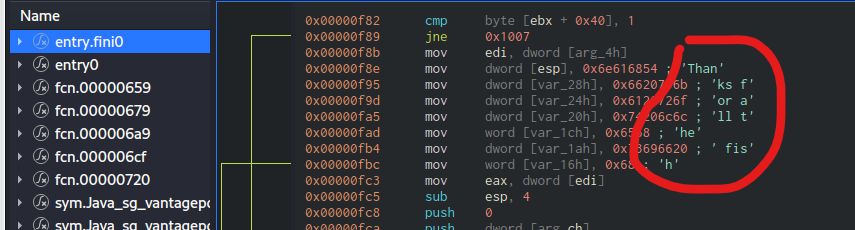


Бачимо що завантажується бібліотека foo

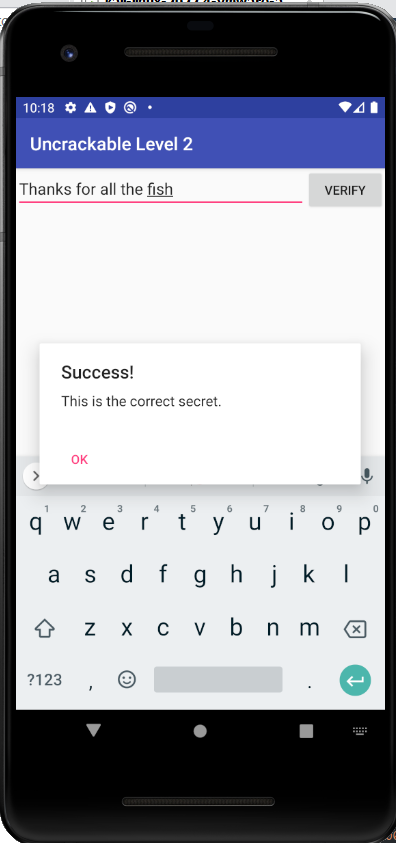


відкриємо її в cutter



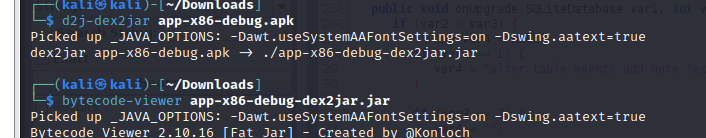


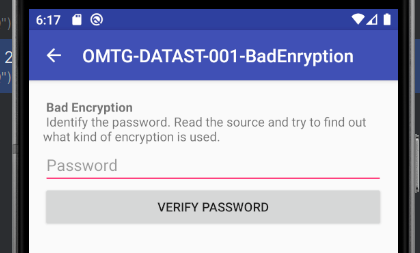
бачимо текст по частинам, спробуємо ввести його



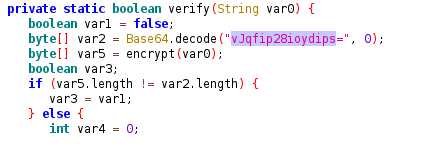
**Hacking Playground:**

встановимо апк <https://github.com/OWASP/MASTG-Hacking-Playground/blob/master/Android/MSTG-Android-Java-App/app/app-x86-debug.apk>, також відкриємо його в bytecode viewer

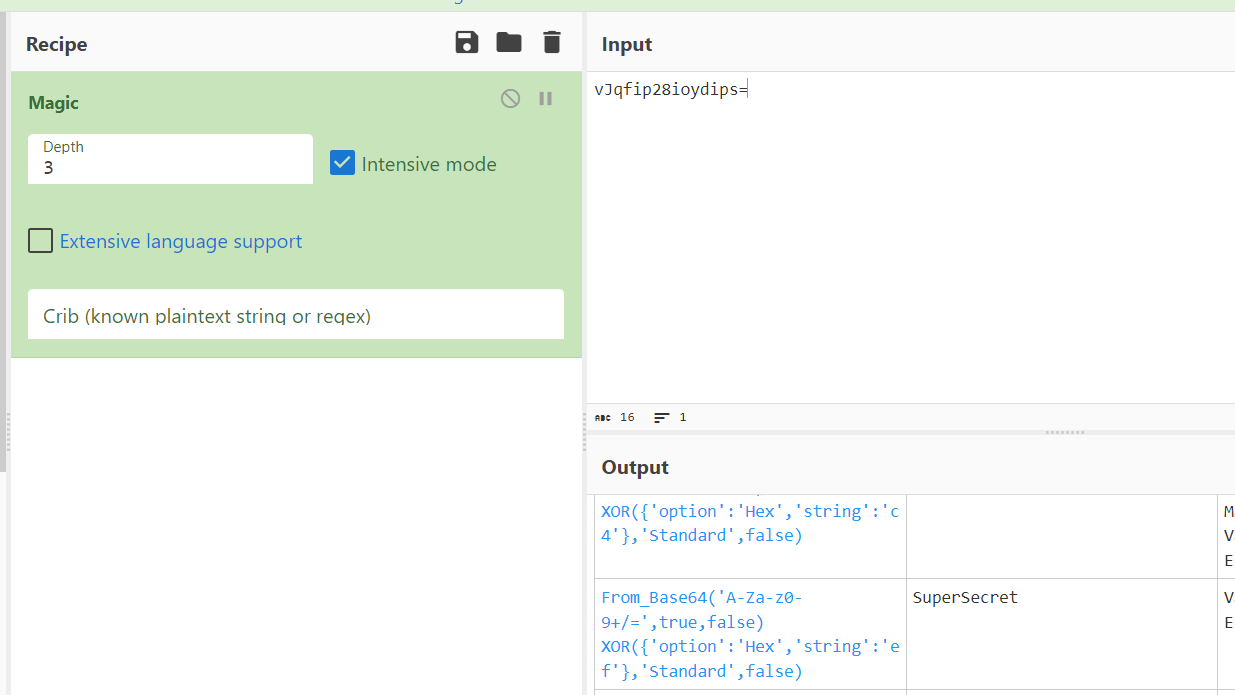


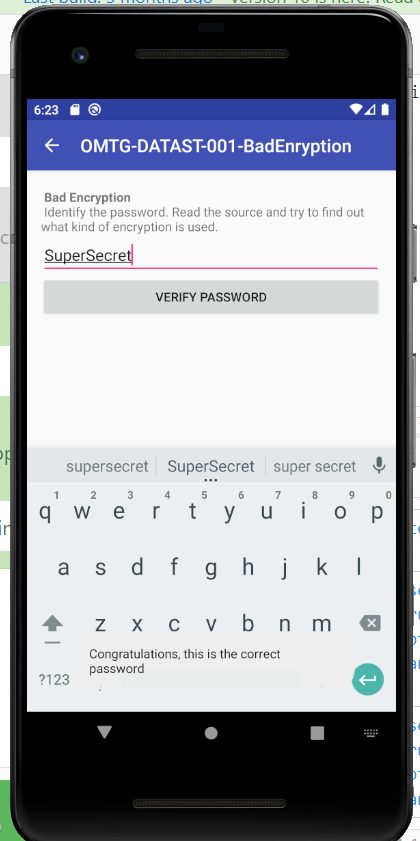






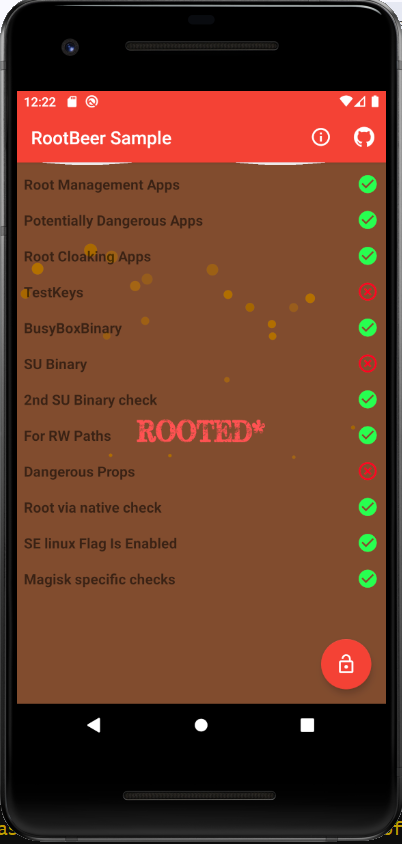
спробуємо розшифрувати цю строку





**Реалiзуйте обхiд перевiрок RootBeer [184] за допомогою Frida [185].**

Запустимо rootbeer та бачимо що він детектує рут



перетворимо на jar файл та через bytecode viever знайдемо функцію, які відповідає за кожен з цих пунктів



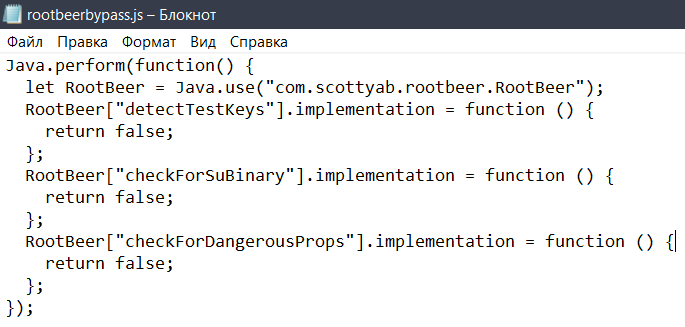
private final List getRootResults() {

return CollectionsKt.listOf(new RootItemResult[]{new RootItemResult("Root Management Apps", this.rootBeer.detectRootManagementApps()), new RootItemResult("Potentially Dangerous Apps", this.rootBeer.detectPotentiallyDangerousApps()), new RootItemResult("Root Cloaking Apps", this.rootBeer.detectRootCloakingApps()), new RootItemResult("TestKeys", this.rootBeer.detectTestKeys()), new RootItemResult("BusyBoxBinary", this.rootBeer.checkForBusyBoxBinary()), new RootItemResult("SU Binary", this.rootBeer.checkForSuBinary()), new RootItemResult("2nd SU Binary check", this.rootBeer.checkSuExists()), new RootItemResult("For RW Paths", this.rootBeer.checkForRWPaths()), new RootItemResult("Dangerous Props", this.rootBeer.checkForDangerousProps()), new RootItemResult("Root via native check", this.rootBeer.checkForRootNative()), new RootItemResult("SE linux Flag Is Enabled", Utils.isSelinuxFlagInEnabled()), new RootItemResult("Magisk specific checks", this.rootBeer.checkForMagiskBinary())});

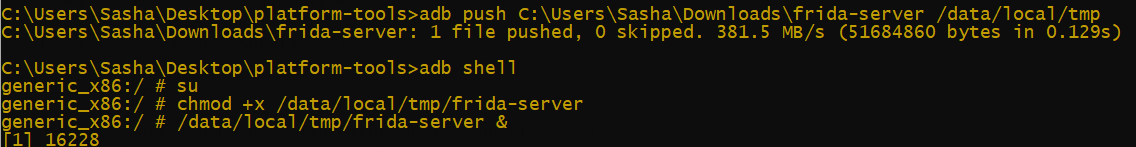
}

У мене детектуються TestKeys, SuBinary та DangerousProps, тому буду працювати з ними

Напишемо функцію яка буде повертати false на ці 3 пункти при виклику rootbeer відповідних функцій



встановимо та запустимо frida-server



Знайдемо PID

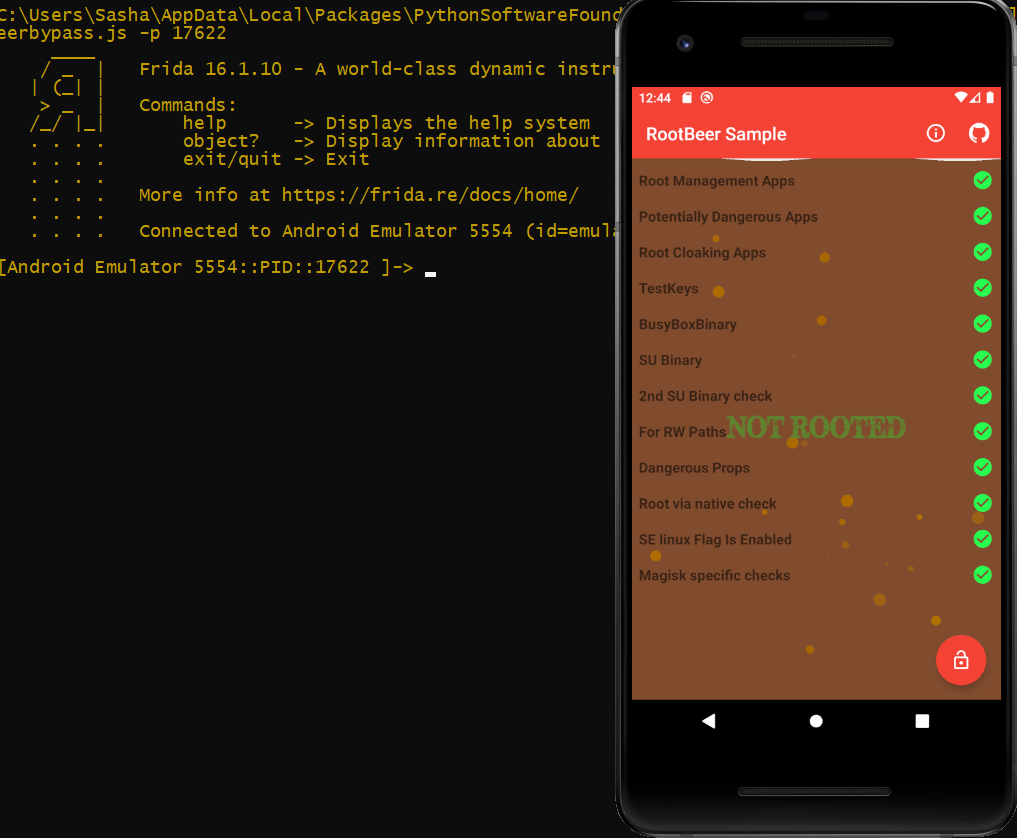




Запускаємо наш код

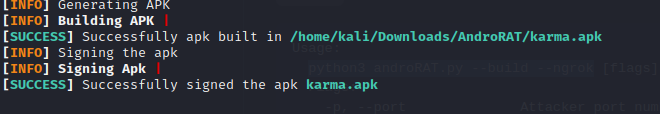
\*\*C:\Users\Sasha\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9\_qbz5n2kfra8p0\LocalCache\local-packages\Python39\Scripts\*\*

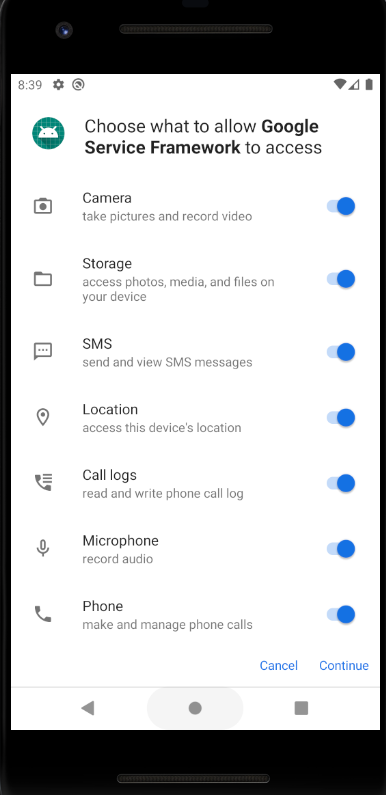
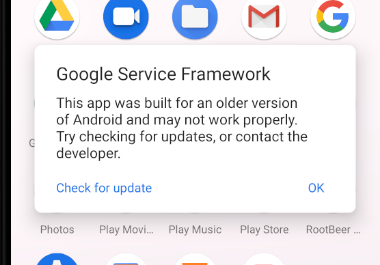
frida -U -l C:\Users\Sasha\Desktop\rootbeerbypass.js -p 17622



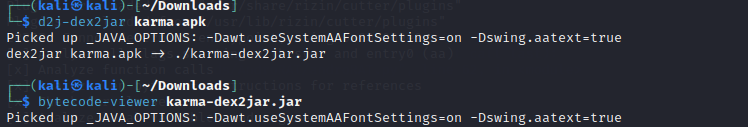
**Дослiдiть зразки систем вiддаленого керування за прикладом роздiлу 8.3.1:**

**Androrat –** [**https://github.com/wszf/androrat**](https://github.com/wszf/androrat)**;**

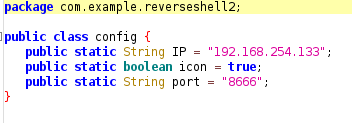
****

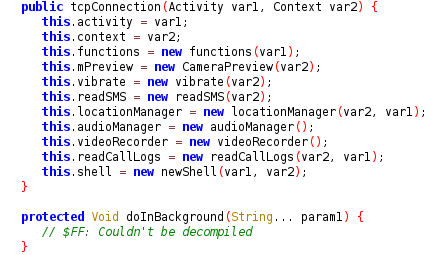
Відкриємо в bytecode viewer



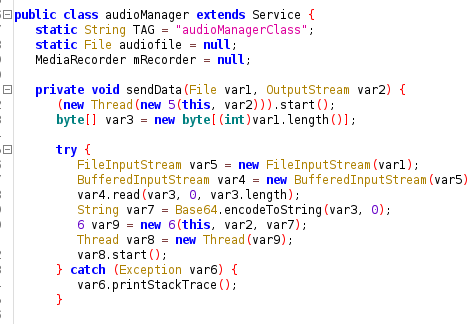
ip та порт які вказані при створенні апк



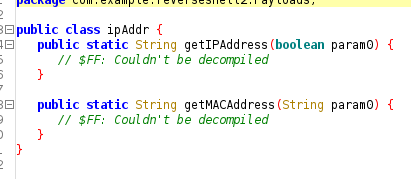
Створення підключення



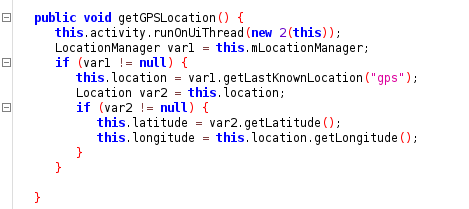
Запис аудіо



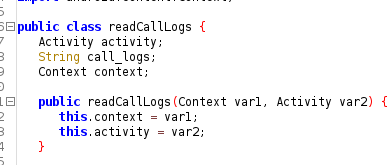
айпі та мак адреса пристрою

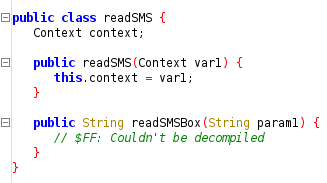


gps

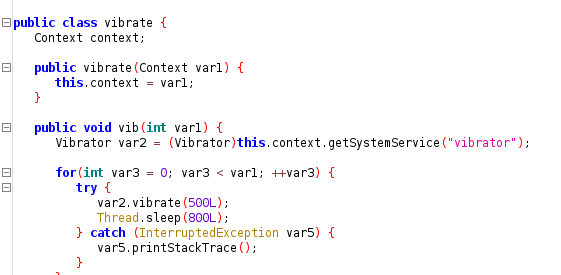


sms та дзвінки





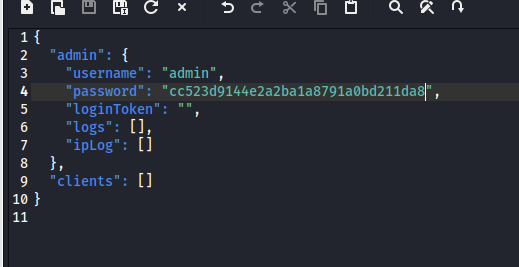
вібрація



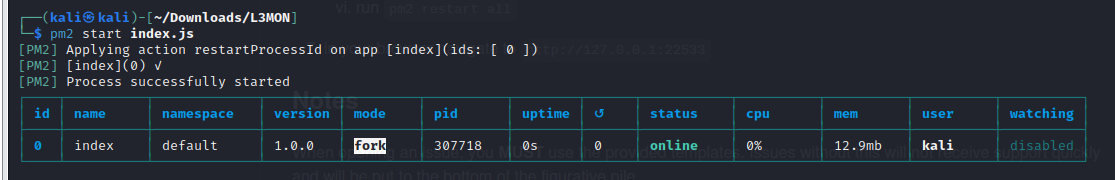
**L3MON –** https://github.com/efxtv/L3MON/tree/main

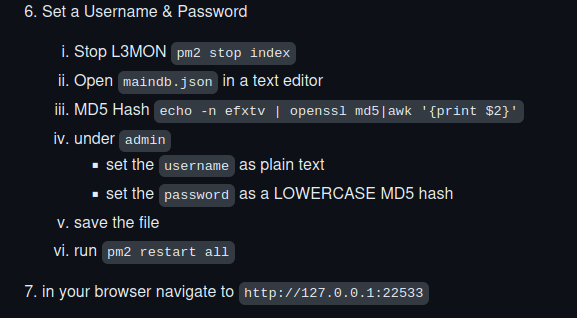
призначимо пароль



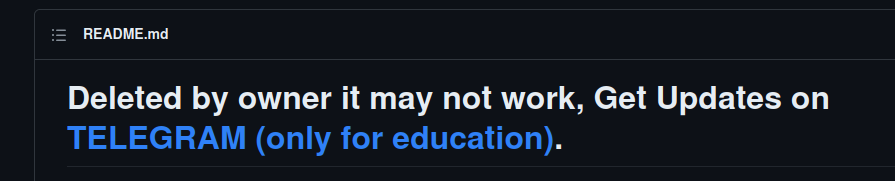


запускаємо сервер





я все зробив по інструкції, але мені видає помилку, тобто неможливо відкрити сервер

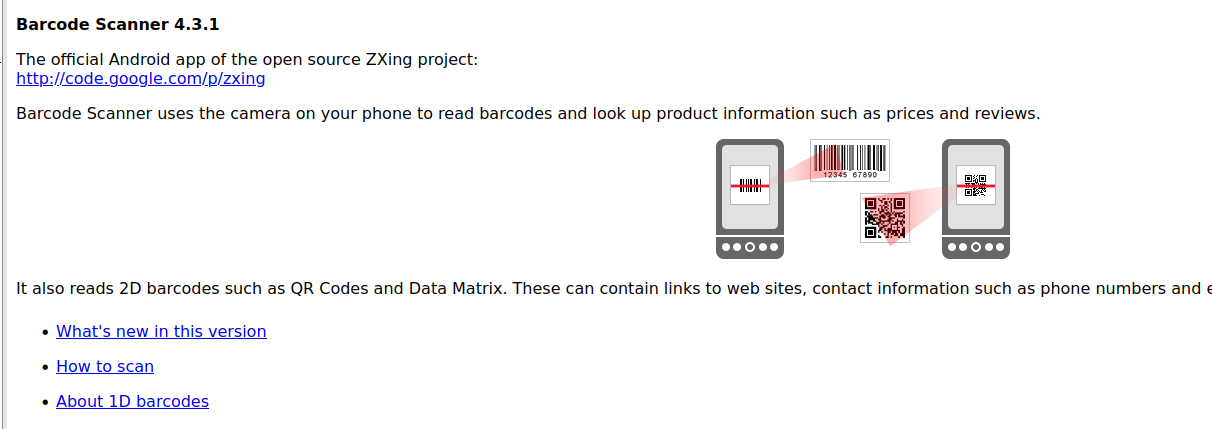


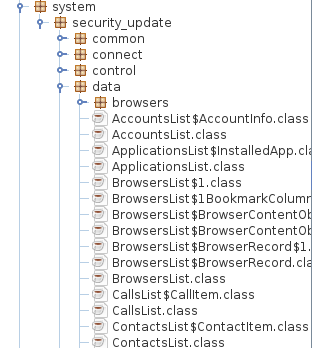
**Проаналiзуйте зразки Monokle [186], за варiантом в табл. 8.1.**

**19 a0c0f4d5ed1e3fb005e4e67bec8629bf, a342b423e0ca57eba3a40311096a4f50**

перетворимо на апк файли та відкриємо в bytecode viewer

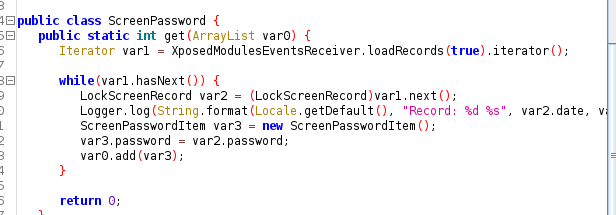
додаток маскується під сканер qr та штрих кодів



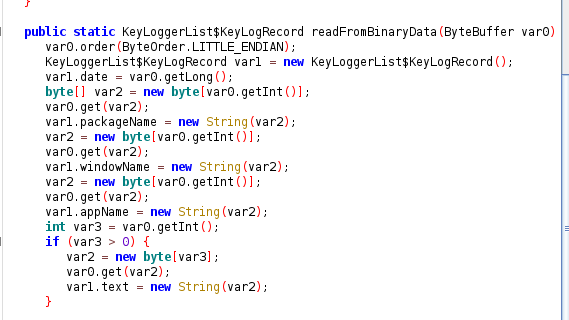


Бачимо класи, які відповідають за шкідливі функції програми

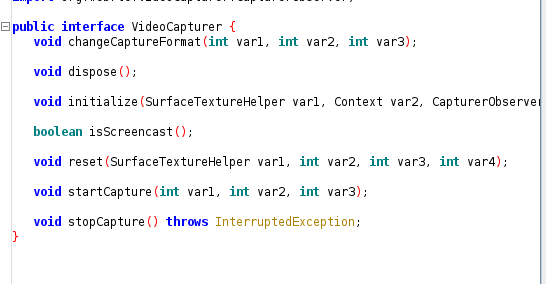
отримання паролю користувача



keylogger

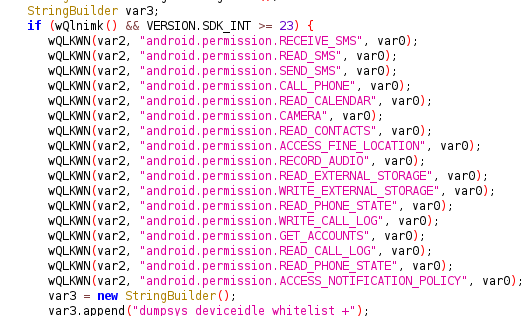


запис відео



другий зразок маскується під signal

запрос дозволів



досуп до повідомлень



в цьому зразку назви шкідливих функції виглядають ось так, тому важко відслідкувати що саме вони роблять

