Analiza podatności aplikacji mobilnych na podstawie OWASP Security Shepherd

26.01.2021

Eryk Winiarz

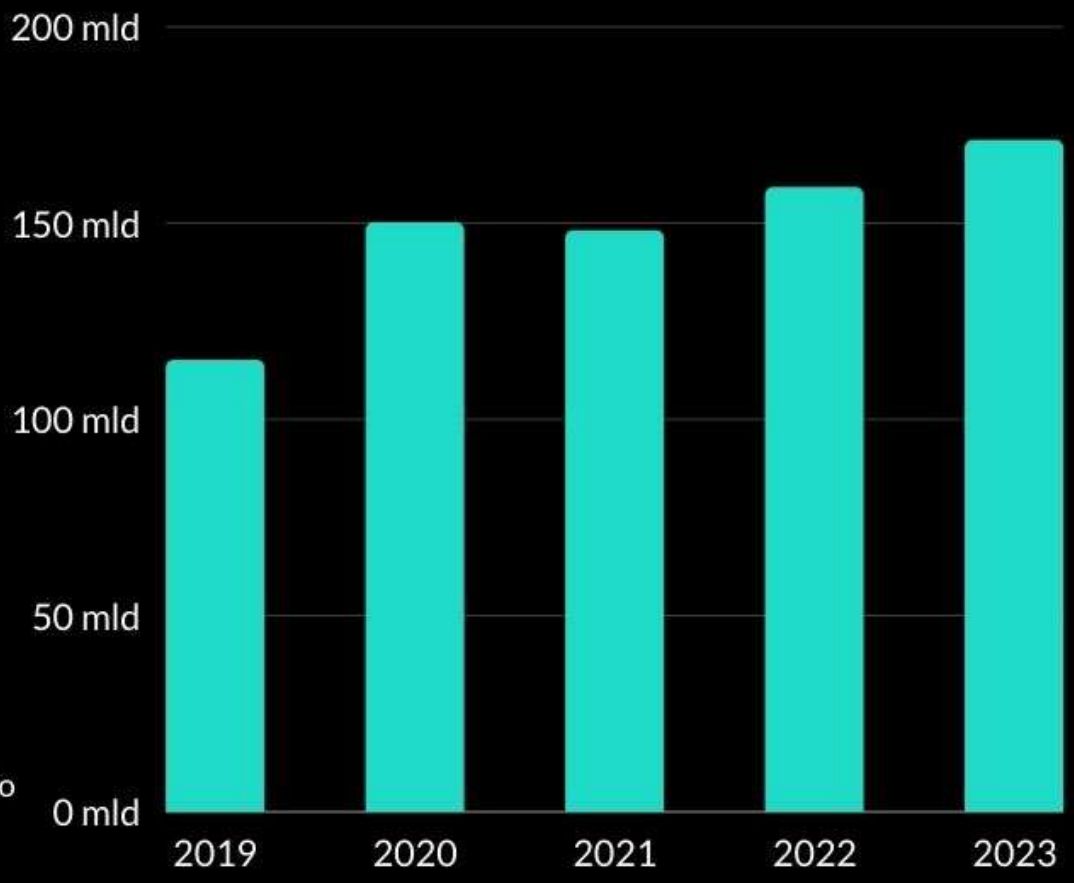
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie | AGH University of Science and Technology

Aplikacje mobilne

Dlaczego bezpieczeństwo aplikacji mobilnych jest tak ważne?

76%

Zgodnie z danymi *Sensor Tower*, około 76% dorosłych osób na świecie korzysta na co dzień ze smartfona.



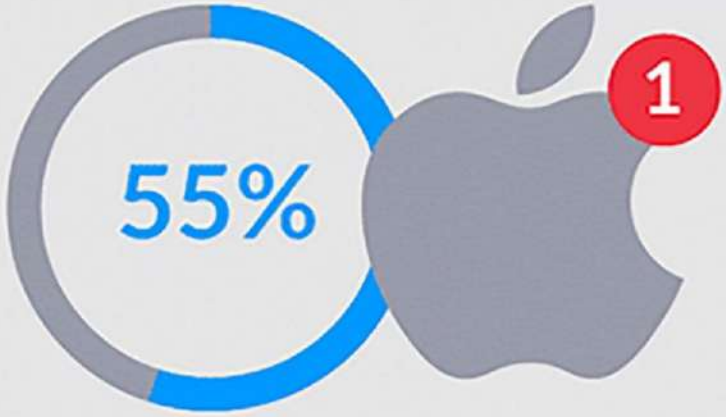
PROGNOZA ILOŚCI POBRAŃ APLIKACJI MOBILNYCH

#1

Ponad trzy czwarte urzadzeń z Androidem pracuje na systemie starszym niż dwa lata

#2

Miesiąc po wypuszczeniu, tylko 55% urządzeń z iOS miało zaintalowaną najnowszą wersję systemu

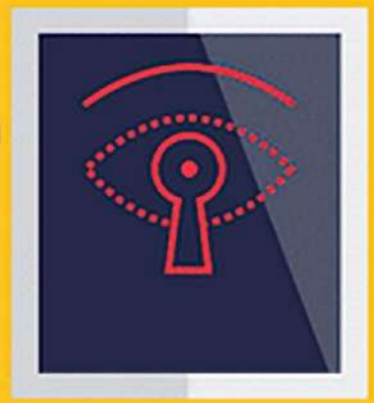


#3

35 procent komunikacji wysyłanej przez urządzenia mobile jest niezaszyfrowana

Ponad 1/3

transmitowanych danych jest zagrożona



#4

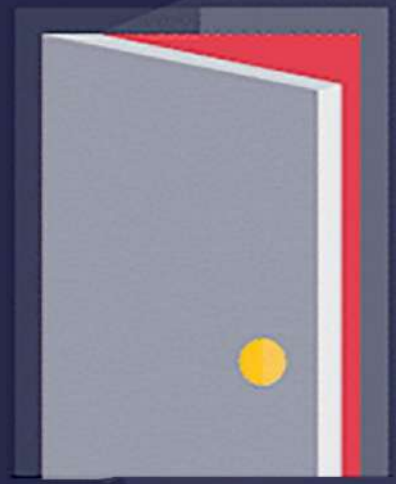
Każde urządzenie mobile łączy się średnio ze 160 różnymi adresami IP każdego dnia

To szkoujące z jak dużą ilością serwisów łączymy się codziennie.

#5

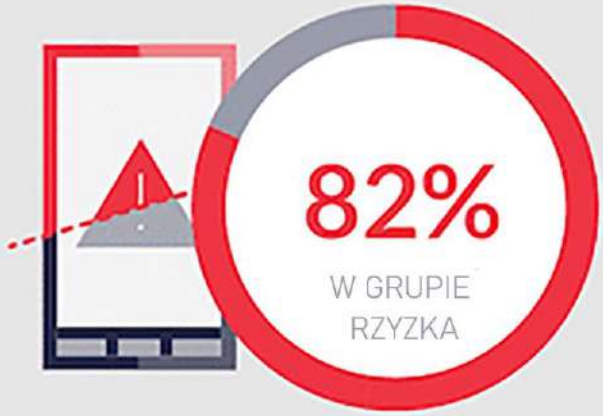
Czterdzieści trzy procent użytkowników urządzeń mobilnych nie posiada hasła, PINu lub wzoru blokującego ich telefony.

Bez hasła, nic nie powstrzymuje osób trzecich przed przejęciem danych ze zgubionego lub skradzonego telefonu.



#6

Osiemdziesiąt dwa procent urządzeń z Androidem było podatnych na co najmniej 25 podatności w systemie operacyjncym Android



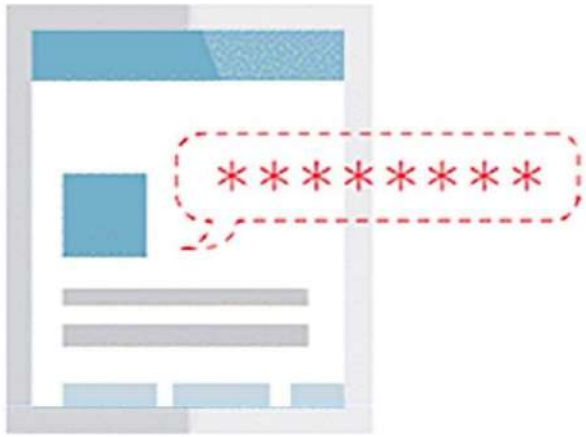
#7

Aplikacje biznesowe są 3 razy bardziej podatne na wyciek danych logowania niż reszta apliacji

Aplikacje biznesowe ujawniają dane osobowe i firmowe

#8

Aplikacje społeczne są trzy razy bardziej podatne na wyciek hasła logowania użytkownika niż pozostałe aplikacje



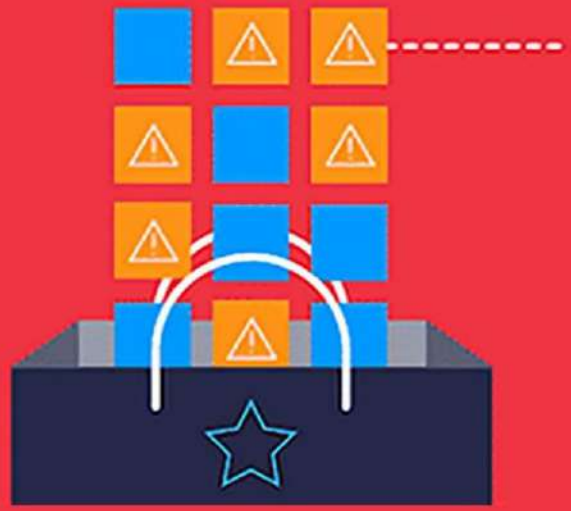
#9

Jedna na cztery aplikacje mobilne mają przynajmniej jedną podatność wysokiego ryzyka.

24.7%

#10

Połowa aplikacji z pobraniami na poziomie od 5 do 10 milionów ma problem z bezpieczeństwem



Im bardziej popularna aplikacje, tym większa szansa na podatność.

Grafiki na podstawie danych z nowsecure.com

Charakterystyka podatności

podatności są to podatności client-side

podatności może być użyta bez fizycznego dostępu dourządzenia

podatności może być wykonana bez praw admina (jailbrak lub root)

Problemy aplikacji mobilnych
Niebezpieczne przechowywanie danych

» Jednym z problemów które często dotykają aplikacji mobilnych jest właśnie przechowywanie danych w sposób, który umożliwia osobom postronnym dostanie się do nich. Atakujący często mogą w bardzo prosty sposób obejść protokoły bezpieczeństwa aplikacji (jeśli autorzy w ogóle jakieś zaimplementują) i wykraść poufne dane, co oprócz samej utraty danych i potencjalnym wykorzystaniu ich, skutkuje na wizerunku i zaufaniu do aplikacji. Spotykanym również problemem bezpieczeństwa jest zabezpieczanie tylko tych „ważnych" elementów aplikacji, pomijając przy tym mniej ważne, np. logi, przez co mimo że np. protokół logowania do aplikacji jest chroniony i nie jesteśmy w stanie z niego nic „wyciągnąć", to już resetowanie hasła i używane do niego wartości są przechowywane w sposób jawny na telefonie (zobaczycie w zadaniu ☺).

Problemy aplikacji mobilnych

Używanie słabych algorytmów kryptograficznych

» Kolejnym problemem często spotykanym w aplikacjach mobilnych, jest używanie przestarzałych, słabych algorytmów szyfrujących. Problem ten w zasadzie wiąże się bezpośrednio z poprzednim, ponieważ używając algorytmów kryptograficznych, które są powszechnie uważane za niebezpieczne i znane są metody ich łamania, możemy doprowadzić do utraty danych i wszystkich tego konsekwencji wymienionych w poprzednim slajdzie. Na szczęście ten problem jest coraz rzadziej spotykany, a producenci urządzeń mobilnych udostępniają własne, bezpieczne i nowoczesne rozwiązania (np. CryptoKit w iOS) do ochrony danych aplikacji.

Problemy aplikacji mobilnych
Podatność na inżynierię odwrotną

» Inżynieria odwrotna to problem który dotyczy nie tylko aplikacji mobilnych, niestety, praktycznie nie da się całkowicie przed nią obronić. Istnieją narzędzia, które w prosty sposób pozwalają z pliku .apk otrzymać cały kod źródłowy aplikacji. Chronić się przed tym możemy na różne sposoby, Google dla przykładu oferuje narzędzie ProGuard, które to posiada różne funkcje pozwalające chronić naszą aplikację – możemy automatycznie zaciemnić, zoptymalizować i zmniejszać (chodzi tu o code shrinking) kod, żeby skutecznie utrudniać (bo 100% bezpieczeństwa nie uzyskamy raczej nigdy) inżynierię odwrotną aplikacji.

Problemy aplikacji mobilnych
Content Provider

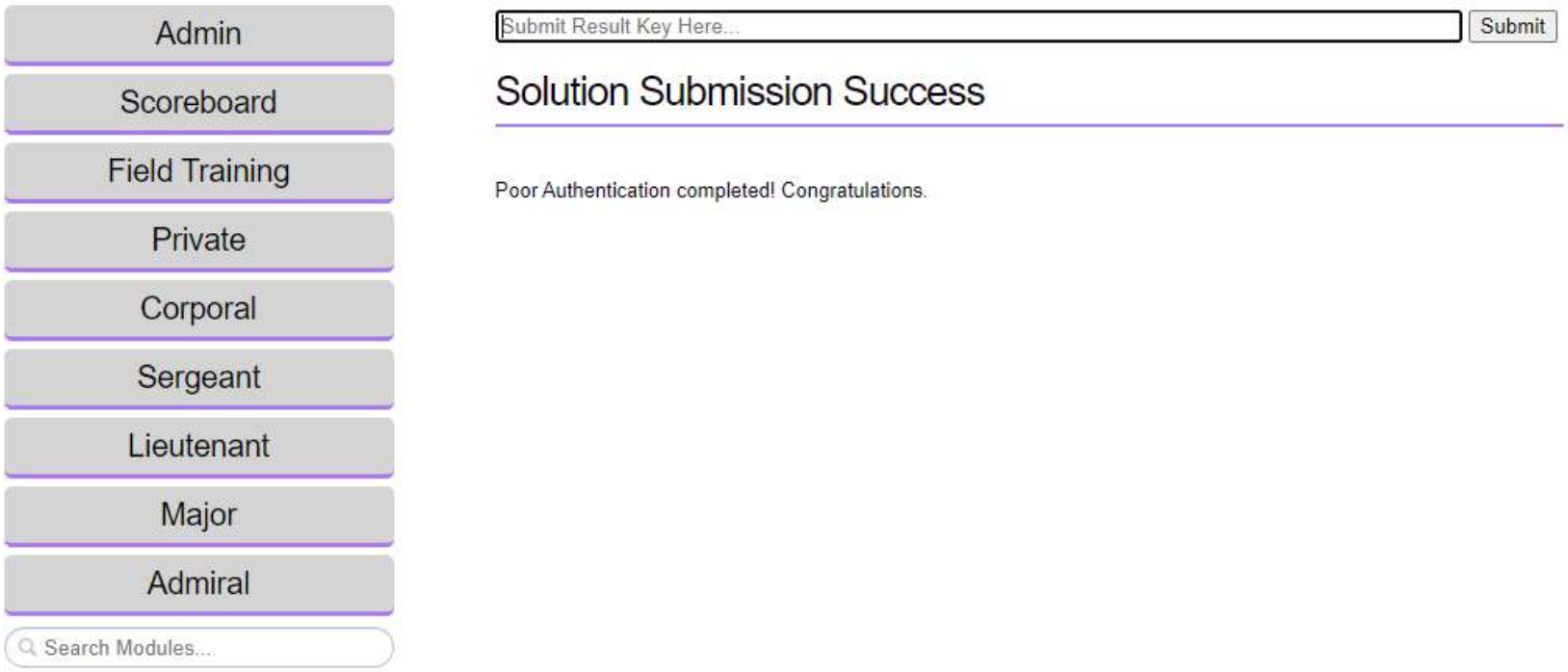
» Zacznijmy od tego, o czym tak naprawdę mówimy. Teoretycznie, dane aplikacji są dostępne tylko dla owej aplikacji. Jeśli aplikacje chcą dzielić się swoimi danymi z innymi aplikacjami, Content Provider to interfejs który do tego służy. Używa on standardowych metod (insert, query, update, delete) żeby uzyskać dostęp do danych aplikacji. Każdy Content Provider, ma swój unikalny URI, zaczynający się wyrażeniem „content://” i aplikacja która zna URI innej, jest w stanie wykonywać działania wymienione wcześniej na danych innej aplikacji. Brzmi to dosyć niebezpieczne – dlatego oczywiście jeśli producent pomyślał o bezpieczeństwie, to nie każda aplikacja jest w stanie tak „atakować” inną – przykładem ochrony jest wymaganie odpowiedniego pozwolenia w pliku AndroidManifest.xml aplikacji.

Problemy aplikacji mobilnych
Content Provider cd.

» Na przykład, do wbudowanej aplikacji do obsługi wiadomości tekstowych systemu Android, możemy uzyskać dostęp poprzez URI „**content://sms/inbox”,** ale aplikacja musi mieć zadeklarowane pozwolenie READ_SMS w wspomnianym wcześniej AndroidManifest.xml, żeby pobrać dane.

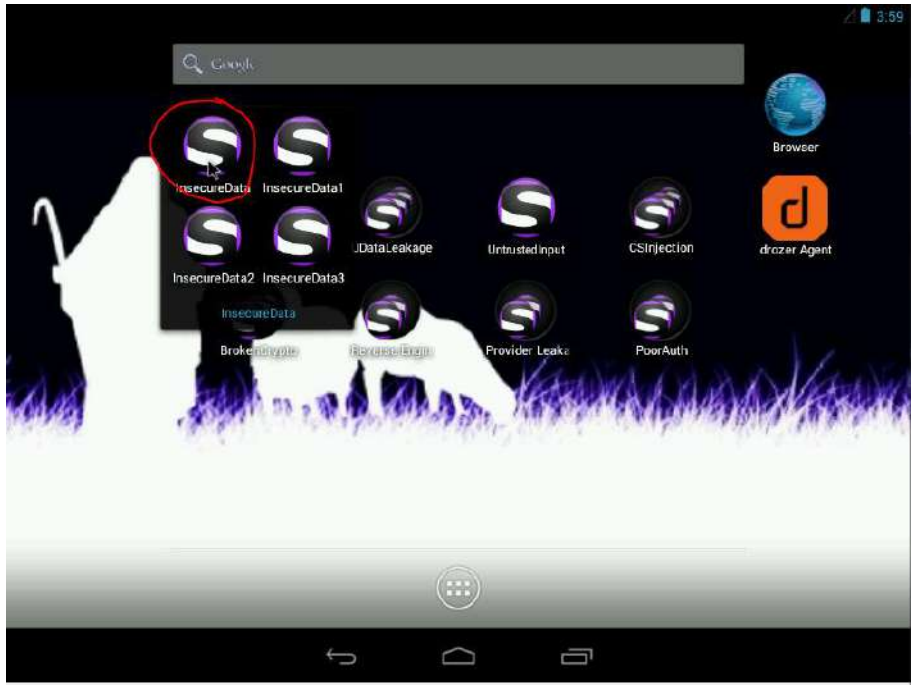
Sprawdzanie poprawności rozwiązania

Rozwiązanie sprawdzamy poprzez wpisanie rozwiązania pod odpowiednim zadaniem i kliknięcie „Submit”



Zadanie 1 – InsecureData – 10 min

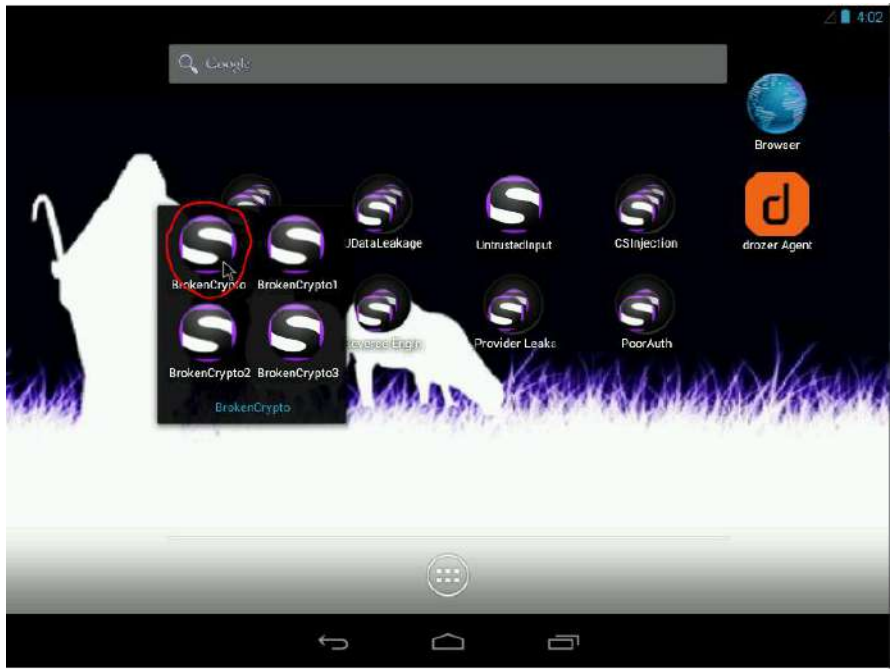
Znalezienie klucza w nieprawidłowo przechowywanych plikach aplikacji



zakładka private -> insecure data storage

Zadanie 2 – BrokenCrypto – 10 minut

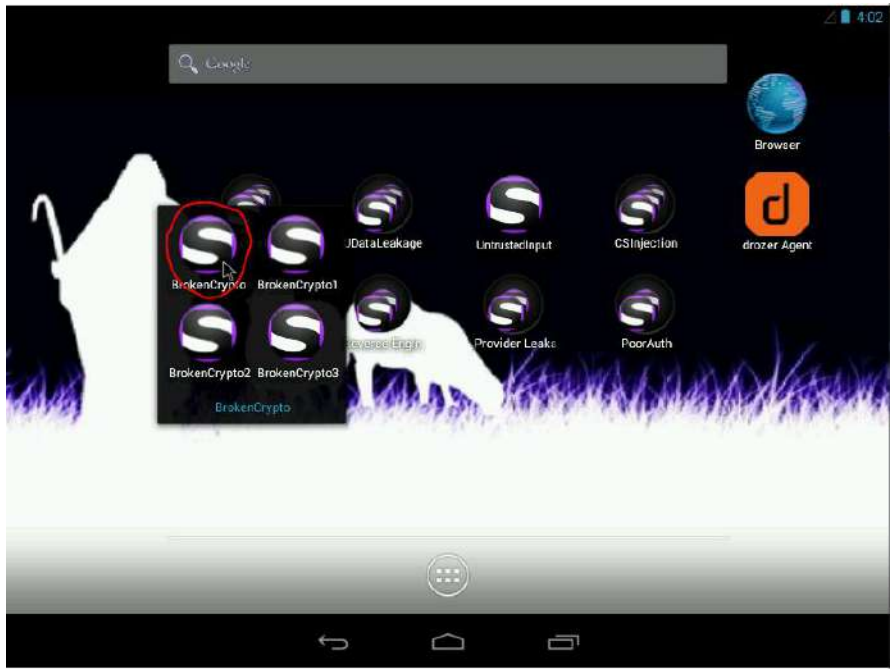
Znalezienie luki w implementacji funkcji czatu



zakładka corporal -> broken crypto

Zadanie 2 – BrokenCrypto – 10 minut

Znalezienie luki w implementacji funkcji czatu
(podpowiedź: jakie cyfry i litery wchodzą w skład wiadomości?)



zakładka corporal -> broken crypto

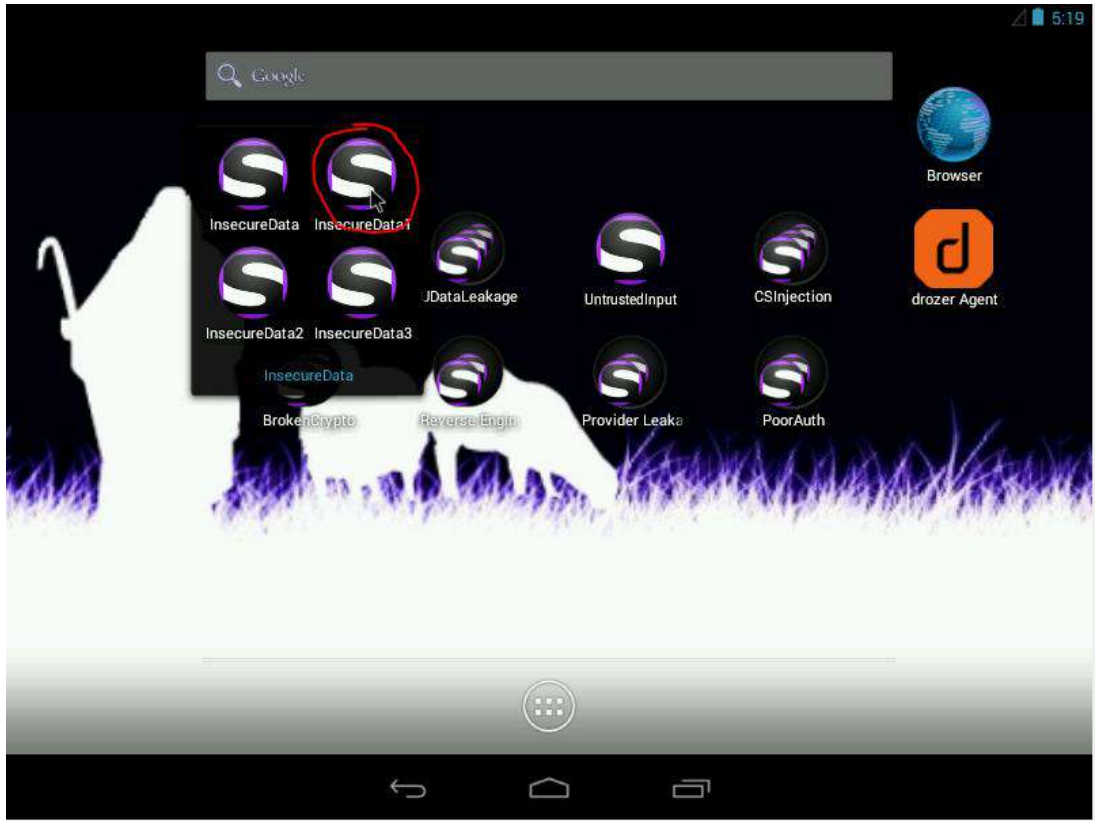
Zadanie 3 – PoorAuthentication – 10 minut

Zresetowanie hasła na podstawie zapisanych logów aplikacji

zakładka corporal -> poor authentication

Zadanie 4 – InsecureData1 – 10 minut

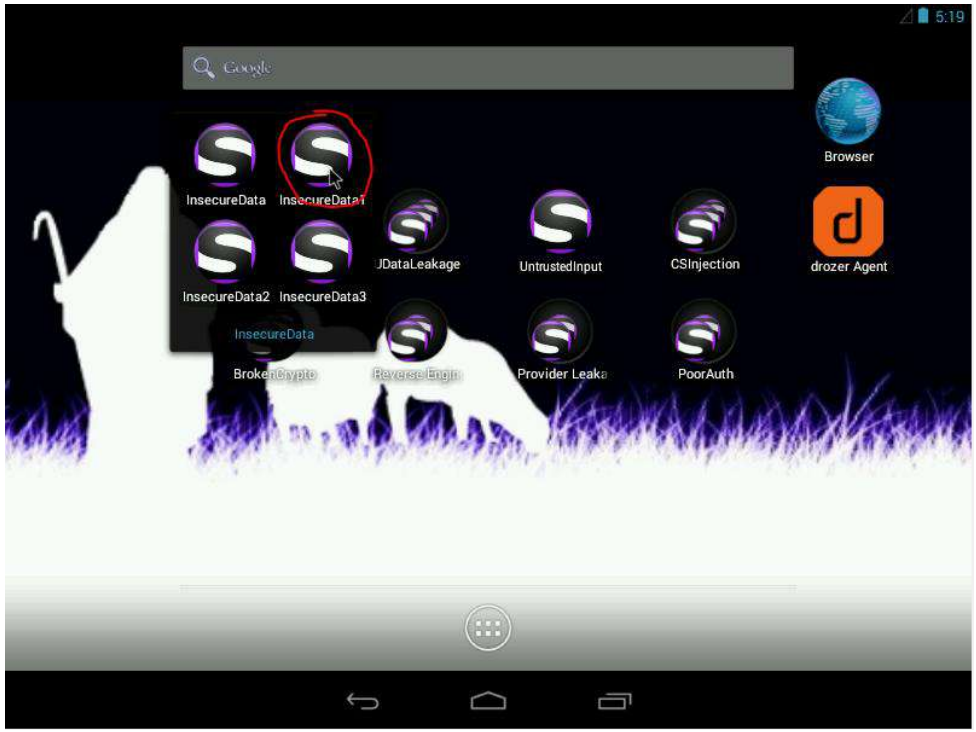
Znalezienie klucza jako hasła administratora



zakładka seargant -> insecure data storage 1

Zadanie 4 – InsecureData1 – 10 minut

Znalezienie klucza jako hasła administratora (podpowiedź: jaki encoding kończy się znakami ==?)

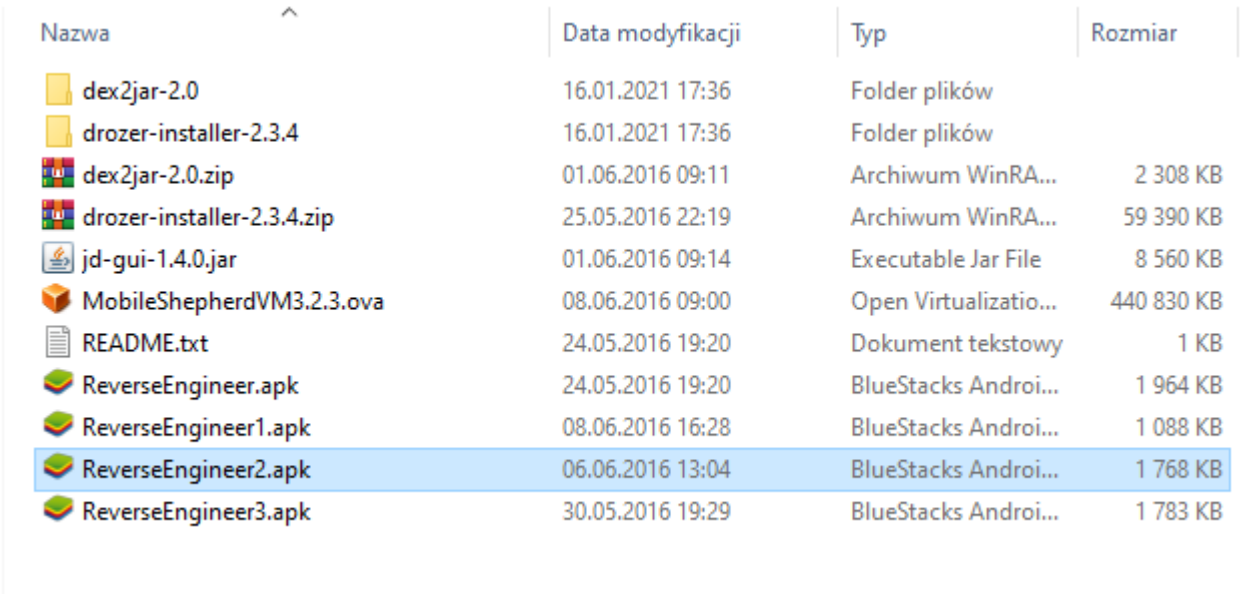


zakładka seargant -> insecure data storage 1

Zadanie 5 – Reverse Engineering – 15 minut

Zrobienie reverse engineeringu

/ To get to the key reverse engineer the APK and find the correct Activity which performs a conditional statement to check the validity of the key. /



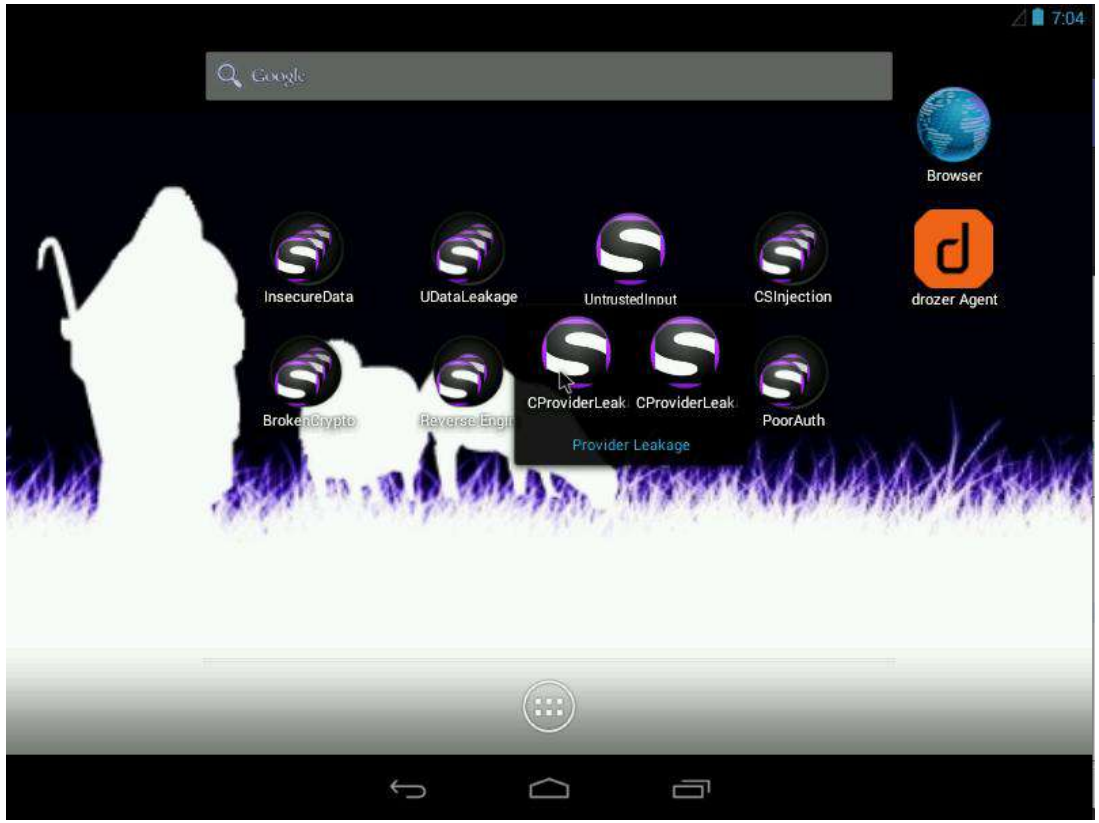
Nazwa	Data modyfikacji	Typ	Rozmiar
dex2jar-2.0	16.01.2021 17:36	Folder plików	
drozer-installer-2.3.4	16.01.2021 17:36	Folder plików	
dex2jar-2.0.zip	01.06.2016 09:11	Archiwum WinRA...	2 308 KB
drozer-installer-2.3.4.zip	25.05.2016 22:19	Archiwum WinRA...	59 390 KB
jd-gui-1.4.0.jar	01.06.2016 09:14	Executable Jar File	8 560 KB
MobileShepherdVM3.2.3.ova	08.06.2016 09:00	Open Virtualizatio...	440 830 KB
README.txt	24.05.2016 19:20	Dokument tekstowy	1 KB
ReverseEngineer.apk	24.05.2016 19:20	BlueStacks Androi...	1 964 KB
ReverseEngineer1.apk	08.06.2016 16:28	BlueStacks Androi...	1 088 KB
ReverseEngineer2.apk	06.06.2016 13:04	BlueStacks Androi...	1 768 KB
ReverseEngineer3.apk	30.05.2016 19:29	BlueStacks Androi...	1 783 KB

Do ćwiczenia używamy dex2jar oraz jd-gui

zakładka corporal-> reverse engineering

Zadanie 6 – CProviderLeakage– 5 minut

Uzyskanie notatki z kluczem z użyciem adb



zakładka private -> CProviderLeakage