Gestionnaires de Mots de Passe

El Bouinbi Mohammed

Plan De Présentation

- Introduction .

- Analyse Statique :
  - Lastpass

- Analyse Dynamique :
  - Lastpass

- Conclusion .

Introduction

Introduction

Un **gestionnaire de mots de passe** est un type de logiciel ou de service en ligne qui permet à un utilisateur de gérer ses mots de passe, soit en centralisant l'ensemble de ses identifiants et mots de passe dans une base de données (portefeuille), soit en les calculant à la demande.

Le Gestionnaire de mots de passe vous permet de :

- créer et enregistrer des mots de passe uniques et sécurisés que vous n'avez pas besoin de retenir .
- protéger tous vos mots de passe enregistrés grâce à la sécurité intégrée .
- saisir automatiquement les mots de passe de votre comptes .

Les raisons d'utiliser un gestionnaire de mots de passe :

- Ne plus jamais oublier son mot de passe.
- Avoir des comptes sécurisés.
- Un accès rapide aux sites.

Analyse Statique

AndroidManifest.xml

Le fichier **manifest** d'application **Android** fournissent des informations à propos de cette application :

- Min SDK Version
- Permissions
- Activities
- Content Providers

Permissions : Définit les données et les composants matériels auxquels les applications ont besoin d'accéder :

- Camera , Contacts , Internet , Read/Write External Storage , Package Management ...

Content Providers : Utilisé pour fournir des données de vos applications à d'autres applications :

- Parfois utilisé pour partager des données entre un groupe d'applications connecté entre eux .

- Si "content providers" = exported , cela peut être très dangereux et exposer les données à tout utilisateur ou applications de l'appareil .

AndroidManifest.xml

Activités : Essentiellement, UI elements ou différents "Screens" dans les applications :

- Camera , Contacts , Internet , Read/Write External Storage , Package Management …

- Certaines activités doivent être protégées :
  + Account Details
  + Money Transfer Screens
  + Hidden Screens ...

- Un activité a exported="True" est accessible en dehors des applications .

Example : LastPass

Permissions : Certaine permission de apps LastPass à partir de AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="504027637" android:versionName="5.4.2.7637"
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="30"/>
    <uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.NFC"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.BIND_ACCESSIBILITY_SERVICE"/>
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
    <uses-permission android:name="com.android.vending.BILLING"/>
    <uses-permission android:name="org.onepf.openiab.permission.BILLING"/>
    <uses-permission android:name="com.sec.android.iap.permission.BILLING"/>
    <uses-permission android:name="android.permission.USE_BIOMETRIC"/>
    <uses-permission android:name="android.permission.PACKAGE_USAGE_STATS"/>
    <uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
    <uses-permission android:name="android.permission.USE_CREDENTIALS"/>
    <permission android:name="com.lastpass.lpandroid.permission.AUTOFILL_AUTH" android:protectionLevel="signature"/>
    <uses-permission android:name="com.lastpass.lpandroid.permission.AUTOFILL_AUTH"/>
    <uses-permission android:name="com.lastpass.authenticator.permission.CLOUD_SYNC_LPA"/>
    <permission android:name="com.lastpass.lpandroid.permission.CLOUD_SYNC_LPM" android:protectionLevel="signature"/>
    <uses-permission android:name="com.lastpass.lpandroid.permission.CLOUD_SYNC_LPM"/>
    <uses-permission android:name="com.lastpass.lpandroid.permission.C2D_MESSAGE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="nu.tommie.inbrowser.PERMISSION_READ_URL"/>
    <uses-permission android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
    <permission android:name="com.lastpass.lpandroid.permission.QUICK_SETTINGS_TILE" android:protectionLevel="signature"/>
```

LastPass

Example : LastPass

- Buckup définit si les données d'application peuvent être sauvegardées et restaurées par un utilisateur .
  Dans notre cas cette propriété est désactiver :

```
android:name="com.lastpass.lpandroid.app.LPApplication" android:allowBackup="false"
```

- Certaines activités doivent être protégées : Account Details
  Dans notre cas pour " account details " on a exported="false" :

```
.service.autofill.WhitelistAppTaskService" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
.service.account.LoginCheckService" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
.service.LogFileDeletionService" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
.service.account.GetMaskedIpJob" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
.service.account.EmergencyAccessShareesUpdaterService" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
.service.BigIconDownloaderJob" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
.service.account.AccountRecoveryDeleteOtpOnServerJob" android:permission="android.permission.BIND_JOB_SERVICE" android:exported="false"/>
```

- Un activité a exported="True" est accessible en dehors des applications .
  Exemple depuis notre apps :

```
android:name="com.google.android.gms.appinvite.PreviewActivity" android:exported="true">
```

LastPass

Exemple : LastPass

- Content Providers : Si "content providers" = exported , cela peut être très dangereux .
  Dans notre cas cette propriété est désactiver :

```
553 /receiver>
561 provider android:name="androidx.core.content.FileProvider" android:exported="false" android:authorities="com.lastpass.lpandroid.fileprovider"
566     <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/file_paths"/>
561 /provider>
```

LastPass

- Conclusion : On peut dire après analyse de fichier AndroidManifest.xml de cette apps que :

  + Les permissions nécessaires pour exécuter l'apps sont des permissions normal :
    Seulement les permissions nécessaires.

  + Apps n'autorisent pas la sauvegarde or "Backup"

  + Apps protège bien les activités , exceptionnel y a des activités a besion d'acceder au autre apps ou l'inverse , donc exported="True" dans cette cas .

Strings.xml

Hardcoded Strings :

- Hardcoded Strings se trouve souvent dans resources/strings.xml .

- Hardcoded Strings peut également être trouvé dans le code source de l'activité .

- On lit le fichier string.xml pour avoir des informations comme :

    + Login bypass (username/password/credentiels)
    + http or https
    + API Keys Exposed
    + Firebase URLs (firebase.io)
    + sql database
    + AWS Feature

Exemple : LastPass

- Login bypass : no login credentiels in strings.xml

- http or https : apps utilise https donc more sécurité

```
href=https://link.lastpass.com/help-account-recovery-Android&gt;Learn more here.&lt;/a&gt;</string>
```

LastPass

- API Keys : Peut être dangereux , car API Keys est utilisé pour l'authentification

```
<string name="google_api_key">AIzaSyCkjXyLDU_Er5yKI-3WtfMhrtg_Q4JH_x0</string>
<string name="google_app_id">1:781750596476:android:c8a8c848350fee53</string>
<string name="google_crash_reporting_api_key">AIzaSyCkjXyLDU_Er5yKI-3WtfMhrtg_Q4JH_x0</string>
```

- Firebase : Peut être dangereux , car on peut énumérer cette URL pour obtenir des informations

```
<string name="firebase_database_url">https://lastpass-147b2.firebaseio.com</string>
<string name="firebase_feature_flags">Firebase feature flags</string>
```

Analyse Statique

(De Source-code ou la Fonctionnalité)

Analyse Statique : De Fonctionnalité

- Dans cette étape on faire une analyse statique de source code afin de vérifier si cette apps gère correctement le mot de passe principal , les clés dérivées Associées et pour déterminer les algorithmes de chiffrement et de déchiffrement , Stockage des clés ...

- Pour effectuer cette étape on a besoin premièrement le source code de cette apps pour l'analyse , pour cela on utilise un logiciel s'appelle Jadx-GUI.

- Jadx-GUI : Command line et GUI Tools pour produire le code source Java à partir de fichiers Android Dex et Apk .

Exemple : LastPass

- A strong Master Password :

Lorsqu'un utilisateur crée un compte pour le gestionnaire de mots de passe LastPass, il crée également un mot de passe principal. Le mot de passe principal est utilisé pour s'authentifier sur le compte LastPass via l'extension de navigateur ou en se connectant à www.lastpass.com.

LastPass

```
public final void q(@NotNull String str, @NotNull byte[] bArr, @NotNull String str2) {
    Intrinsics.e(str, "username");
    Intrinsics.e(bArr, "assembledMasterPassword");
    Intrinsics.e(str2, "authSessionId");
    this.e.K("adfs_auth_session_id", str2);
    String encodeToString = Base64.encodeToString(bArr, 2);
    Intrinsics.d(encodeToString, "Base64.encodeToString(as…Password, Base64.NO_WRAP)");
    AuthenticatorDelegate.DefaultImpls.a(this, new LoginFlow(str, encodeToString, false, 0, 0, null, null, null, false, null, null, false,
}
```

Le mot de passe principal doit être long et unique, avec un mélange de types de caractères - cela a un impact direct sur la sécurité globale des données car d'autres clés de chiffrement sont générées à partir de ce mot de passe.

Exemple : LastPass

- Local-Only Encryption Model :

Le gestionnaire de mots de passe LastPass utilise un cryptage local uniquement. Ce type de solution est conçu pour permettre uniquement à utilisateur de déchiffrer et d'accéder à ses données. Ce qui signifie que toutes les données sensibles sont chiffrées et déchiffrées exclusivement sur la machine locale de l'utilisateur (comme Chrome, Firefox, iPhone, Android, le Web, etc.).

LastPass

```java
/* loaded from: classes.dex */
public class AFKeystoreWrapper {
    private Context AFInAppEventParameterName;
    public KeyStore AFKeystoreWrapper;
    public final Object valueOf = new Object();
    public String values = "";
    public int AFInAppEventType = 0;

    public AFKeystoreWrapper(Context context) {
        this.AFInAppEventParameterName = context;
        AFLogger.valueOf("Initialising KeyStore..");
        try {
            KeyStore instance = KeyStore.getInstance("AndroidKeyStore");
            this.AFKeystoreWrapper = instance;
            instance.load(null);
        } catch (IOException | KeyStoreException | NoSuchAlgorithmException | CertificateException e) {
            AFLogger.values("Couldn't load keystore instance of type: AndroidKeyStore", e);
        }
    }

    private static boolean valueOf(String str) {
        return str.startsWith("com.appsflyer");
    }

    public final String AFInAppEventParameterName() {
        String str;
        synchronized (this.valueOf) {
            str = this.values;
        }
        return str;
    }
```

```java
    /* JADX INFO: Access modifiers changed from: package-private */
    public final List<Map<String, Object>> AFInAppEventParameterName() {
        for (w wVar : this.onInstallConversionDataLoadedNative.values()) {
            wVar.AFKeystoreWrapper(this.onInstallConversionFailureNative, true);
        }
        Map<w, Map<String, Object>> map = this.onInstallConversionFailureNative;
        if (map == null || map.isEmpty()) {
            return new CopyOnWriteArrayList(Collections.emptyList());
        }
        return new CopyOnWriteArrayList(this.onInstallConversionFailureNative.values());
    }
}
```

Exemple : LastPass

- Account Lockout :

LastPass protège également contre les attaques par force brute en verrouillant les comptes après plusieurs tentatives infructueuses de connexion. Nous surveillons régulièrement les comptes pour détecter des signes d'activité irrégulière ou suspecte et suspendrons automatiquement les comptes le cas échéant.

LastPass

```java
package com.nulabinc.zxcvbn.guesses;

import com.nulabinc.zxcvbn.matchers.Match;

/* loaded from: classes2.dex */
public class BruteforceGuess extends BaseGuess {
    @Override // com.nulabinc.zxcvbn.Guess
    public double a(Match match) {
        double pow = Math.pow(10.0d, (double) match.a());
        if (Double.isInfinite(pow)) {
            pow = Double.MAX_VALUE;
        }
        return Math.max(pow, match.a() == 1 ? 11.0d : 51.0d);
    }
}
```

Exemple : LastPass

- AES 256-bit Encryption :

LastPass crypte les données utilisateur avec l'algorithme de confiance Advanced Encryption Standard (AES) en mode Cipher Block Chaining (CBC) avec une clé de 256 bits générée à partir du mot de passe principal de chaque utilisateur.

```java
/* JADX INFO: Access modifiers changed from: package-private */
@Nullable
@RequiresApi
public static BiometricPrompt.CryptoObject a() {
    try {
        KeyStore instance = KeyStore.getInstance("AndroidKeyStore");
        instance.load(null);
        KeyGenParameterSpec.Builder b = Api23Impl.b("androidxBiometric", 3);
        Api23Impl.d(b);
        Api23Impl.e(b);
        KeyGenerator instance2 = KeyGenerator.getInstance("AES", "AndroidKeyStore");
        Api23Impl.c(instance2, Api23Impl.a(b));
        instance2.generateKey();
        Cipher instance3 = Cipher.getInstance("AES/CBC/PKCS7Padding");
        instance3.init(1, (SecretKey) instance.getKey("androidxBiometric", null));
        return new BiometricPrompt.CryptoObject(instance3);
    } catch (IOException | InvalidAlgorithmParameterException | InvalidKeyException | KeyStoreException | NoSuchAlgorithmException
        Log.w("CryptoObjectUtils", "Failed to create fake crypto object.", e);
        return null;
    }
}
```

LastPass

Exemple : LastPass

- Key Derivation :

LastPass utilise par défaut 100100 tours de PBKDF2-SHA256 pour créer la clé de chiffrement .
PBKDF2 est : Derived Key = PBKDF2(PRF, Password, Salt, Iterations, Key Length)

LastPass

```java
public class Pbkdf2JniWrapper {

    /* renamed from: a  reason: collision with root package name */
    private boolean f5274a;

    @Inject
    public Pbkdf2JniWrapper() {
        try {
            System.loadLibrary("lastpass_pbkdf2");
            this.f5274a = true;
        } catch (UnsatisfiedLinkError e) {
            LpLog.v(e);
            e.printStackTrace();
        }
    }

    private final native byte[] pbkdf2HmacSha256Impl(byte[] bArr, byte[] bArr2, int i, int i2);

    public final boolean a() {
        return this.f5274a;
    }

    @NotNull
    public final synchronized byte[] b(@NotNull byte[] bArr, @NotNull byte[] bArr2, int i, int i2) {
        Intrinsics.e(bArr, "password");
        Intrinsics.e(bArr2, "salt");
        return pbkdf2HmacSha256Impl(bArr, bArr2, i, i2);
    }
}
```

Exemple : LastPass

- Conclusion :

Pour conclure LastPass est utilisé les meilleurs et les plus complique algorithmes de cryptage excite maintenant .
Donc il offre un bon niveau de sécurité pour les utilisateur et contre les attaques comme Bruteforce and MITM attaques ...
Car Même si il y a une MITM attack , l'attacker ne peut jamais déchiffrer les donnes crypter à cause de ces algorithmes .

LastPass

Analyse Dynamique

(Bypass SSL Pinning)

Bypass SSL pinning

- SSL pinning :

Dans les applications mobiles, SSL Pinning ou HTTP Public Key Pinning (HPKP) fournit une couche de sécurité supplémentaire aux communications HTTPS pour éviter, par exemple, les attaques de l'homme du milieu. Il fonctionne côté client et vérifie le certificat du serveur en comparant les hachages des clés publiques pré-groupées avec l'application mobile.

- Bypass SSL pinning :

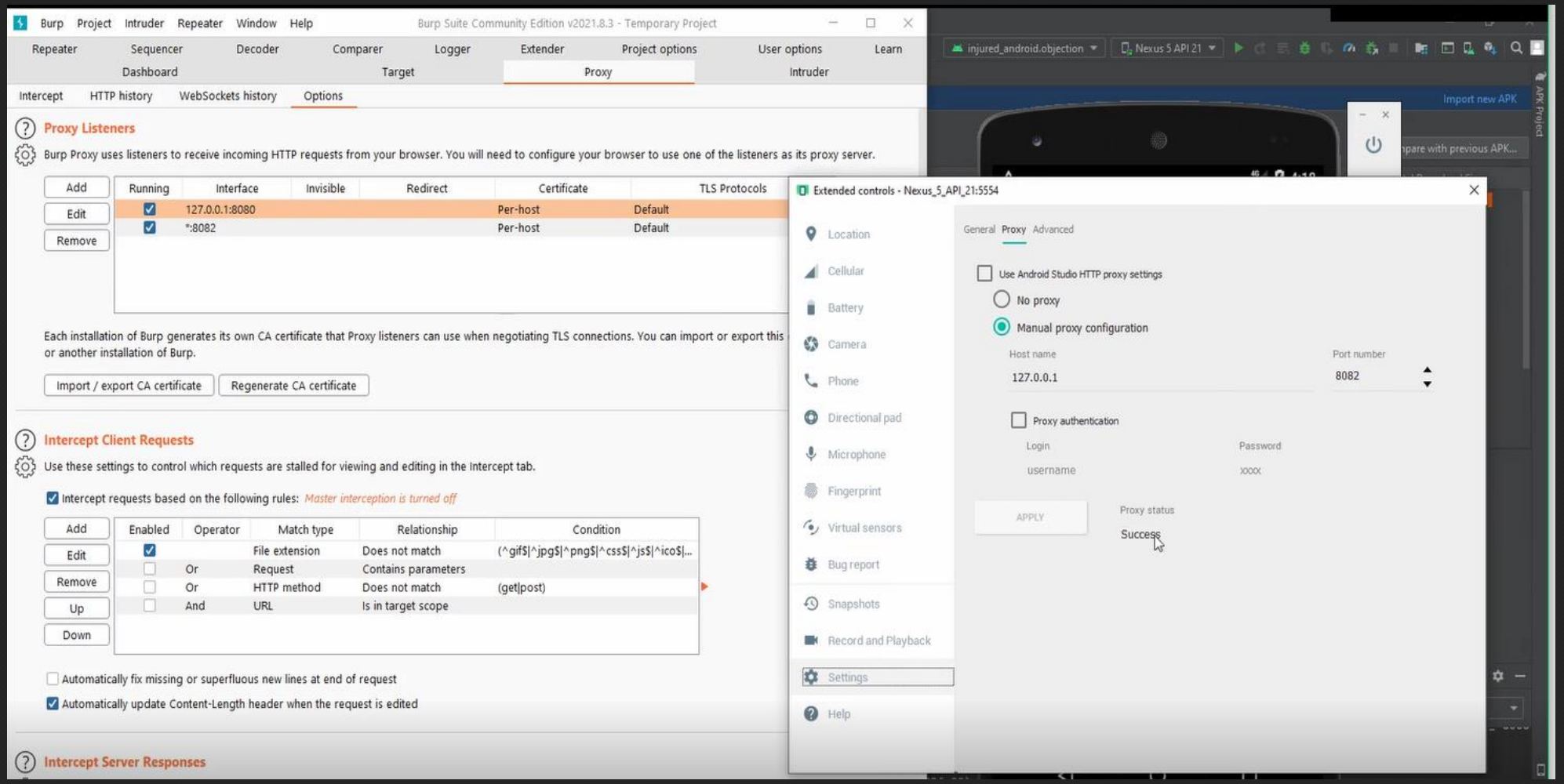
Bypass SSL pinning c'est d'être capable d'intercepter la connexion entre apps et le serveur web .

- Burpsuite :

Un outil dédié à l'audit de sécurité des plateformes web. la fonction principale est un proxy web qui nous permet intercepter la connexion . Ce logiciel est développé par l'entreprise PortSwigger.

Exemple : Bypass SSL pinning

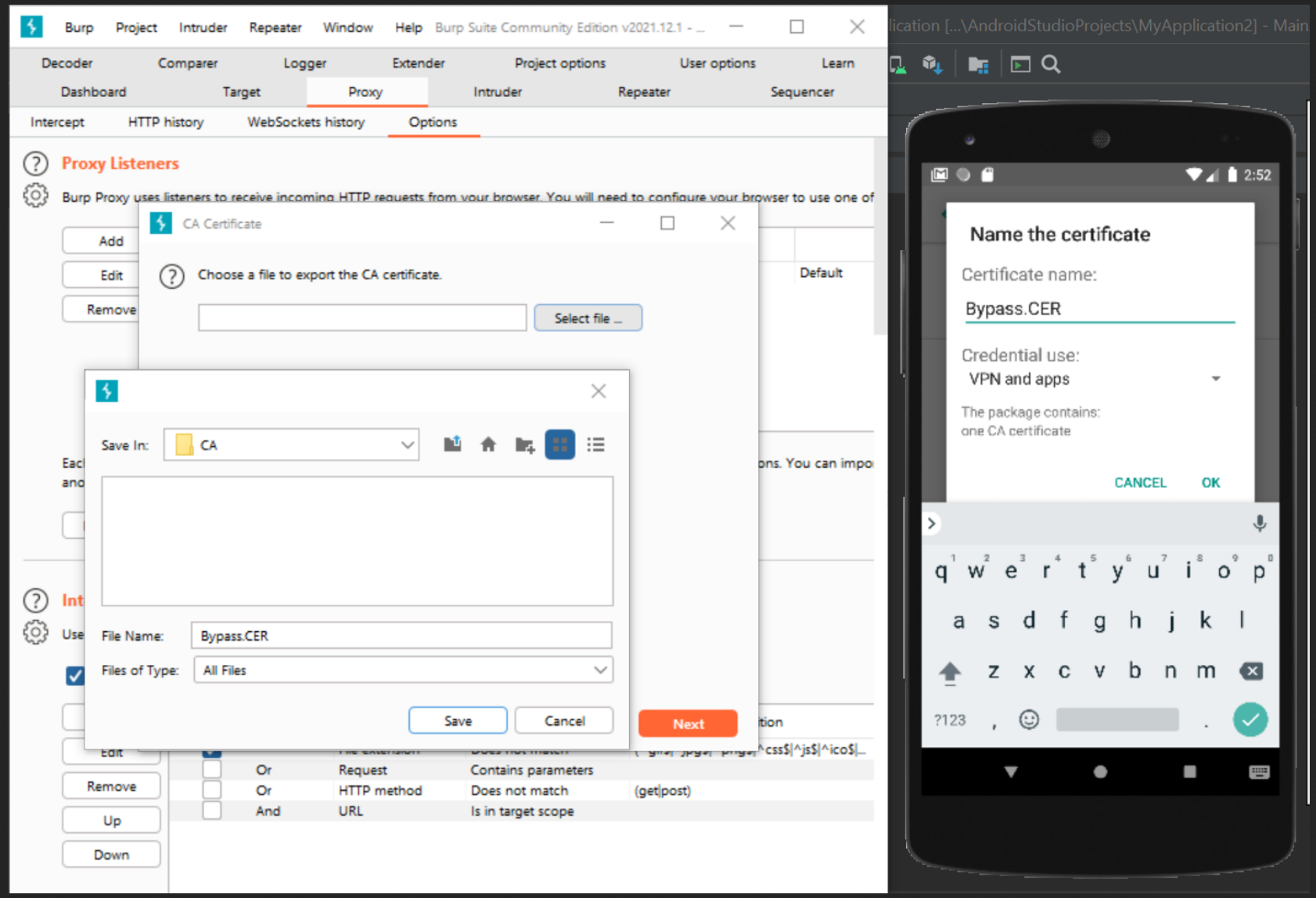
1- La configuration de Burpsuite / émulateur :



LastPass

Exemple : Bypass SSL pinning

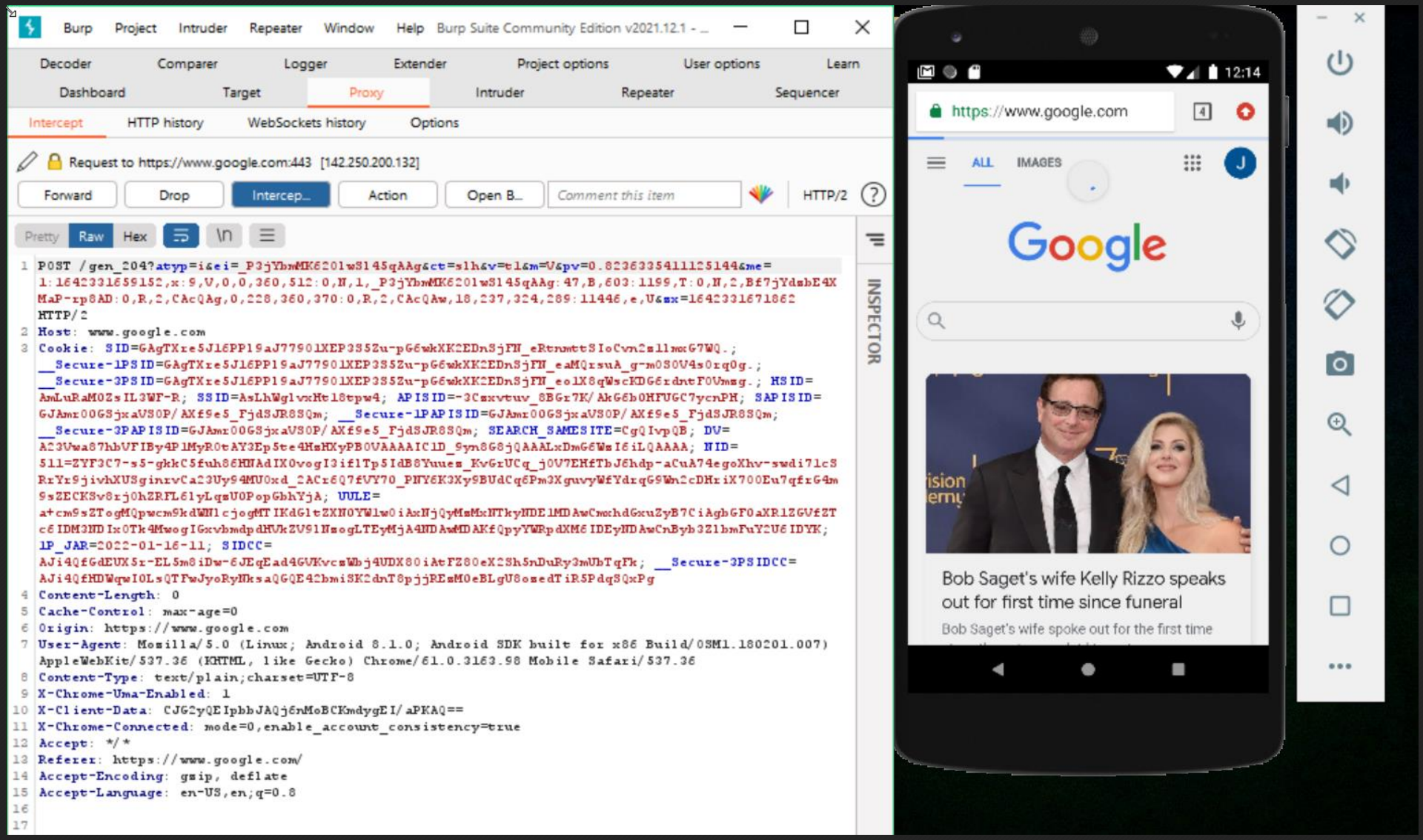
1- L'installation de CA :



LastPass

Exemple : Bypass SSL pinning

1- L'interception de la connexion :



LastPass

Analyse Dynamique

(Injection Frida/Objection)

Injection Frida/Objection

- Frida :

C'est une boîte à outils d'instrumentation de code dynamique. Elle permet d'injecter du code dans le processus analysé pour modifier son état interne et récupérer son état à partir d'un processus maître extérieur.

- Objection :

Est un outil d'exploration dynamique qui fonctionne avec Frida , conçue pour vous aider à évaluer la sécurité de vos applications mobiles, sans avoir besoin d'un jailbreak.

- Première étape : injection de frida manuellement dans notre apps : lastpass

Exemple : Injection frida

- Nous avons 2 options pour injecter Frida's Gadget dans l'application Android :

  Méthode 1 : si l'APK ciblé contient une bibliothèque native (<apk>/lib/arm64-v8a/libfromapk.so), vous pouvez alors injecter libfrida-gadget.so en tant que dépendance dans la bibliothèque native.

  Méthode 2 : si l'APK ne contient pas de bibliothèque native, vous pouvez injecter le bytecode System.loadLibrary.

Dans notre projet on a utilisé la 1ere méthode .

Les étapes de l'injection :

1- Obtenez le fishier APK de l'application que vous souhaitez tester, par ex. lastpass.apk.

Exemple : Injection frida

2- Utilisez apktool pour décoder l'APK . De préférence sa dernière version.

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ ll
total 41596
-rw-r--r-- 1 joms kali 22392288 Jan 16 06:20 frida-gadget-15.1.14-android-arm64.so
-rw------- 1 joms kali 20332740 Jan 16 06:20 lastpass-password-manager-5-4-2-7637.apk

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ apktool d -rs lastpass-password-manager-5-4-2-7637.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.5.0-dirty on lastpass-password-manager-5-4-2-7637.apk
I: Copying raw resources...
I: Copying raw classes.dex file...
I: Copying raw classes2.dex file...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
I: Copying META-INF/services directory

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ LL
zsh: command not found: LL

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ ll                                                                    127 ×
total 41600
-rw-r--r-- 1 joms kali 22392288 Jan 16 06:20 frida-gadget-15.1.14-android-arm64.so
drwxr-xr-x 9 joms kali     4096 Jan 16 06:26 lastpass-password-manager-5-4-2-7637
-rw------- 1 joms kali 20332740 Jan 16 06:20 lastpass-password-manager-5-4-2-7637.apk
```

Exemple : Injection frida

3- Copiez frida-gadget dans le répertoire APK décompressé.

Il faut copier frida-gadgetin dans architecture arm64-v8a pour injecter après, avant cela, il faut télécharger frida-gadget pour arm64-v8a .

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ ll                                                                    127 ×
total 41600
-rw-r--r-- 1 joms kali 22392288 Jan 16 06:20 frida-gadget-15.1.14-android-arm64.so
drwxr-xr-x 9 joms kali     4096 Jan 16 06:26 lastpass-password-manager-5-4-2-7637
-rw------- 1 joms kali 20332740 Jan 16 06:20 lastpass-password-manager-5-4-2-7637.apk

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ cp frida-gadget-15.1.14-android-arm64.so lastpass-password-manager-5-4-2-7637/lib/arm64-v8a/
libfrida-gadget.so

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ cd lastpass-password-manager-5-4-2-7637/lib/arm64-v8a

┌──(joms㉿joms7x)-[~/…/lastpass/lastpass-password-manager-5-4-2-7637/lib/arm64-v8a]
└─$ ll
total 24276
-rw-r--r-- 1 joms kali 22392288 Jan 16 06:32 libfrida-gadget.so
-rw-r--r-- 1 joms kali  2261384 Jan 16 06:26 liblastpass_pbkdf2.so
-rw-r--r-- 1 joms kali   324464 Jan 16 06:26 liblastpass.so
-rw-r--r-- 1 joms kali    10064 Jan 16 06:26 libtool-checker.so
```

Exemple : Injection frida

4- Injection frida-gadget .
pour cela , nous utilisons une bibliothèque python appellee " lief " et plus prècisèment le script au-dessous :

```python
#!/usr/bin/env python3

import lief

libnative = lief.parse("/home/kali/Desktop/lastpass/lastpass-password-manager-5-4-2-7637/lib/arm64-v8a/liblastpass.so")
libnative.add_library("libfrida-gadget.so") # Injection!
libnative.write("/home/kali/Desktop/lastpass/lastpass-password-manager-5-4-2-7637/lib/arm64-v8a/liblastpass.so")
```

5- Vérification d'injection :

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ readelf -d lastpass-password-manager-5-4-2-7637/lib/arm64-v8a/liblastpass.so

Dynamic section at offset 0x53000 contains 30 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libfrida-gadget.so]
 0x0000000000000001 (NEEDED)             Shared library: [liblog.so]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so]
 0x0000000000000001 (NEEDED)             Shared library: [libm.so]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so]
 0x000000000000000e (SONAME)             Library soname: [liblastpass.so]
```

Exemple : Injection frida

4- Re-pack APK :

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ apktool b lastpass-password-manager-5-4-2-7637 -o new_lastpass.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.5.0-dirty
I: Copying lastpass-password-manager-5-4-2-7637 classes.dex file...
I: Copying lastpass-password-manager-5-4-2-7637 classes2.dex file...
I: Checking whether resources has changed...
I: Copying raw resources...
I: Copying libs... (/lib)
I: Copying libs... (/kotlin)
I: Copying libs... (/META-INF/services)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ ll
total 70172
-rw-r--r--  1 joms kali 22392288 Jan 16 06:20 frida-gadget-15.1.14-android-arm64.so
drwxr-xr-x 10 joms kali     4096 Jan 16 06:47 lastpass-password-manager-5-4-2-7637
-rw-------  1 joms kali 20332740 Jan 16 06:20 lastpass-password-manager-5-4-2-7637.apk
-rw-r--r--  1 joms kali 29253314 Jan 16 06:47 new_lastpass.apk
-rw-r--r--  1 joms kali      335 Jan 16 06:42 script.py
```

Exemple : Injection frida

7- Signez l'APK mis à jour en utilisant nous propres clés et zipalign .
Android exige que tous les fichiers APK soient signés numériquement avec un certificat avant d'être installés sur un appareil ou mis à jour.

- creat own keys :

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ keytool -genkey -v -keystore last.keystore -alias lastkey -keyalg RSA -keysize 2048 -validity 10000
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  last
What is the name of your organizational unit?
  [Unknown]:  last
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=last, OU=last, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  y

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
        for: CN=last, OU=last, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
[Storing last.keystore]
```

Exemple : Injection frida

- sign apk :

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ jarsigner -sigalg SHA1withRSA -digestalg SHA1 -keystore last.keystore -storepass lastkey new_lastpass.apk lastkey
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
jar signed.

Warning:
The signer's certificate is self-signed.
The SHA1 algorithm specified for the -digestalg option is considered a security risk. This algorithm will be disabled in
 a future update.
The SHA1withRSA algorithm specified for the -sigalg option is considered a security risk. This algorithm will be disable
d in a future update.

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ jarsigner -verify new_lastpass.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

jar verified.

Warning:
This jar contains entries whose certificate chain is invalid. Reason: PKIX path building failed: sun.security.provider.c
ertpath.SunCertPathBuilderException: unable to find valid certification path to requested target
This jar contains entries whose signer certificate is self-signed.
The SHA1 digest algorithm is considered a security risk. This algorithm will be disabled in a future update.
The SHA1withRSA signature algorithm is considered a security risk. This algorithm will be disabled in a future update.
```

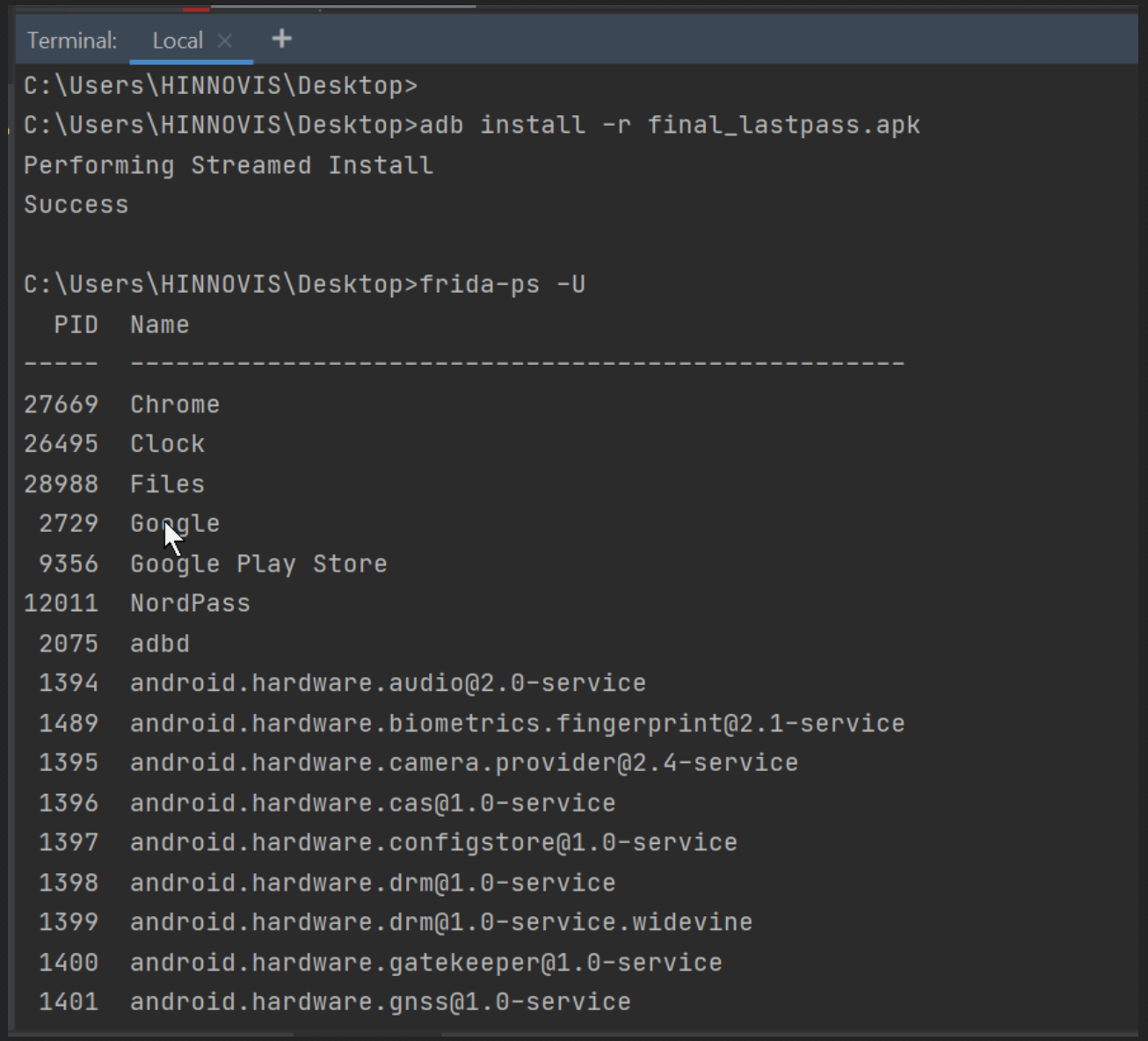
- zipalign apk :

```
┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ zipalign 4 new_lastpass.apk final_lastpass.apk

┌──(joms㉿joms7x)-[~/Desktop/lastpass]
└─$ ll
total 99016
-rw-r--r--  1 joms kali 29392707 Jan 16 07:09 final_lastpass.apk
```

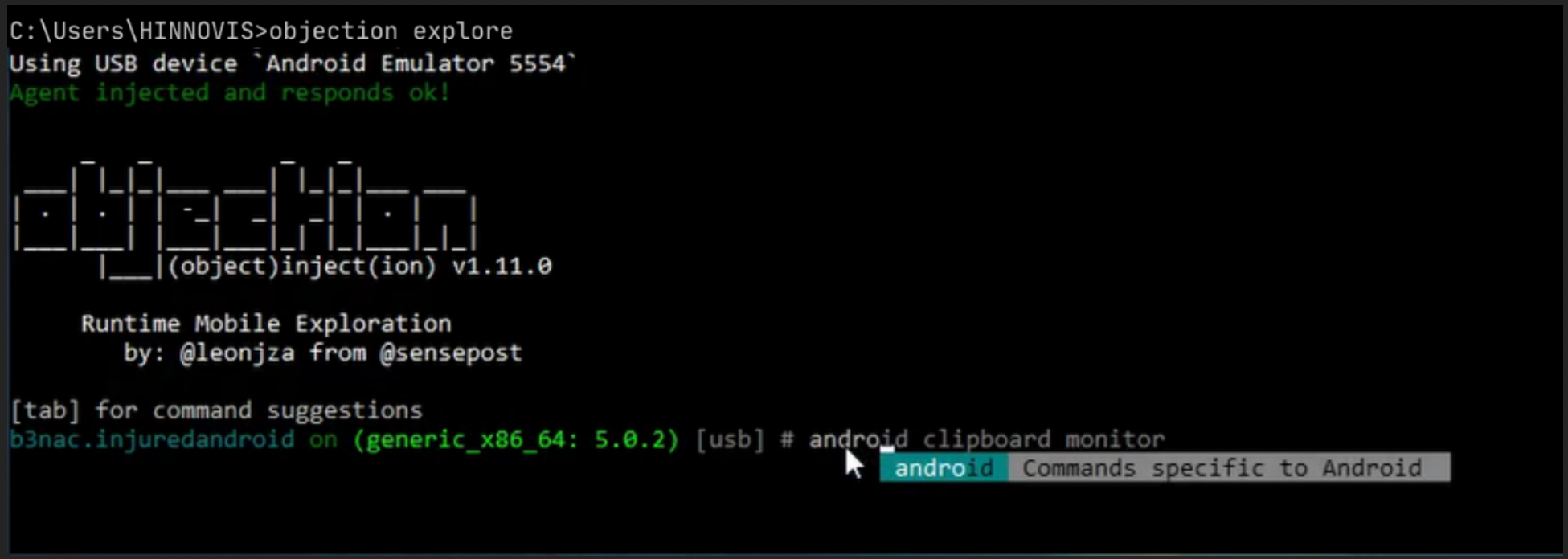
Exemple : Injection frida

8- Installez l'APK mis à jour sur un appareil : On utilise adb pour installer apk



Exemple : Utilisation d'Objection

- Deuxième étape : Apres l'injection de frida on passe à la 2eme étape : c'est utilisation d'objection avec frida lorsque apps est en cours d'exécution .



Conclusion

Conclusion

Pour conclure :

Ce projet nous permet de savoir comment analyser les applications des gestionnaires de mots de posse et bien appliquer le concept de reverse engineering .

Au niveau de l'analyse statique on savoir comment comprendre les fonctionnalités d'application de gestionnaire de mot de passe , le recherche dans les deux fichier Android Manifest.xml et string.xml et comment avoir les point vulnérables dans ces 2 fichiers .

Au niveau de analyse dynamique en savoir comment faire le bypass de certificat SSL et comment intercepter le trafique réseau qui sort de l'application avec le Burpsuite , on comprend aussi l'utilisation de Frida et objection pour injecter plusieurs commande pendant l'exécution .