

# Gestionnaires de mots de passe

ANALYSE STATIQUE & DYNAMIQUE

Cyber Sécurité | Sécurité Mobile

Encadré par :

MR. KOULALI MOHAMMED AMINE

Réalisé par :

YOUNES TASRA

Année Universitaire : 2022-2023

## *Résumé*

---

La cybersécurité est devenue l'un des domaines les plus importants en constante évolution dans l'informatique et l'industrie technologique. Une sécurité défectueuse a entraîné des pertes immenses pour l'économie mondiale. Souvent, l'écueil de ces pertes financières est dû à la sécurité des mots de passe, puisqu'ils constituent la première ligne de défense contre les cybermenaces. C'est la raison pour laquelle il est important de créer et de stocker des mots de passe forts et sécurisés. Cependant, il peut être difficile de se souvenir de nombreux mots de passe différents, et c'est là qu'interviennent les gestionnaires de mots de passe. Ces outils permettent le stockage et la récupération d'informations sensibles à partir d'une base de données chiffrée, en offrant la possibilité de générer des mots de passe aléatoires pour chaque compte.

Le but de ce projet est de décompiler et d'analyser cinq gestionnaires de mots de passe populaires sur Android (Keeper, NordPass, LastPass, Bitwarden et 1Password) afin de vérifier s'il gèrent correctement le mot de passe principale (Master Password) ainsi que les clés dérivées associées.

## *Abstract*

---

Cybersecurity has become one of the most important and constantly evolving areas in the IT and technology industry. Faulty security has resulted in immense losses to the global economy. Oftentimes, the pitfall in such financial loss is due to the security of passwords, since they are considered the first line of defense against cyber threats. This is the reason why it is important to create and store strong and secure passwords. Password managers allow the storage and retrieval of sensitive information from an encrypted database. Users rely on them to provide better security guarantees against trivial exfiltration than alternative ways of storing passwords, such as an unsecured flat text file.

The purpose of this project is to decompile and analyse five popular password managers on Android platform (Keeper, NordPass, LastPass, Bitwarden and 1Password) to see if they correctly handle the Master Password and the associated derived keys.

# Table des matières

<b>Résumé .....</b>	<b>1</b>
<b>Abstract .....</b>	<b>1</b>
<b>Tables des figures .....</b>	<b>6</b>
<b>Introduction .....</b>	<b>7</b>
<b>Documentation .....</b>	<b>8</b>
Notions techniques .....	8
<i>Analyse statique</i> .....	8
<i>Analyse dynamique</i> .....	8
<i>APK</i> .....	8
Environnement de travail .....	9
<i>Kali Linux</i> .....	9
<i>Genymotion</i> .....	9
Outils utilisés.....	10
<i>Apktool</i> .....	10
<i>Android Debug Bridge</i> .....	10
<i>Dex2Jar</i> .....	11
<i>Burpsuite</i> .....	11
<i>JADX</i> .....	11
<i>OpenSSL</i> .....	12
<i>Objection</i> .....	12
<i>Frida</i> .....	12
<b>APP1 - Keeper .....</b>	<b>13</b>
Analyse statique .....	13
<i>Permissions de l'application</i> .....	13
<i>Protection des données</i> .....	14
<i>PBKDF2</i> .....	14
<i>Chiffrement AES-256-GCM</i> .....	14
<i>Chiffrement client</i> .....	14
<i>Record-level encryption</i> .....	15
<i>Two-Factor Authentication</i> .....	15
<i>Algorithme TOTP</i> .....	16

## Table des matières

<i>Identification biométrique</i> .....	17
<i>Vérification du certificat du serveur WebView</i> .....	17
<i>PKCS5/PKCS7</i> .....	17
Analyse dynamique .....	18
<i>Configuration de Burp Suite</i> .....	18
<i>Interception du trafic</i> .....	21
<i>Signup</i> .....	21
<i>Vault login</i> .....	23
<i>Security question</i> .....	23
Conclusion .....	24
<b>App2 - NordPass</b> .....	25
Analyse statique .....	25
<i>Permissions de l'application</i> .....	25
<i>Protection des données</i> .....	26
<i>Utilisation du presse-papiers (Clipboard)</i> .....	26
<i>Certificate Pinning</i> .....	27
<i>Stockage externe</i> .....	27
<i>Root detection</i> .....	28
<i>Algorithmes cryptographiques non sécurisés</i> .....	28
<i>Générateurs de nombres aléatoires non sécurisés</i> .....	29
Analyse dynamique .....	30
<i>SSL Pinning Bypass</i> .....	30
<i>Interception du trafic</i> .....	31
<i>Lancement de l'application</i> .....	31
<i>Signup</i> .....	32
<i>Création du Master Password</i> .....	33
<i>Création d'un mot de passe</i> .....	34
Conclusion .....	35
<b>App3 - LastPass</b> .....	36
Analyse statique .....	36
<i>Permissions de l'application</i> .....	36
<i>Protection des données</i> .....	37
<i>Algorithmes cryptographiques non sécurisés</i> .....	38

## Table des matières

<i>Stockage externe .....</i>	39
<i>Générateurs de nombre aléatoires non sécurisés .....</i>	39
<i>Root detection.....</i>	40
<i>SQLite database .....</i>	40
<i>Certificate Pinning.....</i>	41
<i>Divulgation des adresses IP.....</i>	42
<i>SafetyNet API.....</i>	42
Analyse dynamique .....	42
<i>SSL Pinning Bypass.....</i>	42
<i>Interception du trafic.....</i>	44
<i>Signup.....</i>	44
<i>Se connecter à LastPass.....</i>	45
Conclusion .....	46
<b>App4 - Bitwarden .....</b>	<b>47</b>
Analyse statique .....	47
<i>Permissions de l'application.....</i>	47
<i>Protection des données.....</i>	48
<i>Générateurs de nombres aléatoires non sécurisés.....</i>	49
<i>SQLite Database .....</i>	49
<i>Utilisation du presse-papiers (Clipboard).....</i>	49
<i>Hardcoded sensitive information .....</i>	50
Analyse dynamique .....	50
<i>SSL Pinning Bypass.....</i>	50
<i>Interception du trafic.....</i>	52
<i>Signup.....</i>	52
<i>Se connecter à Bitwarden.....</i>	52
Conclusion .....	53
<b>App5 - 1Password .....</b>	<b>54</b>
Analyse statique .....	54
<i>Permissions de l'application.....</i>	54
<i>Protection des données.....</i>	55
<i>Générateurs de nombres aléatoires non sécurisés.....</i>	56
<i>Divulgation des adresses IP.....</i>	56

## Table des matières

<i>Root detection</i> .....	56
<i>Algorithmes cryptographiques non sécurisés</i> .....	57
<i>Certificate Pinning</i> .....	58
<i>Stockage externe</i> .....	58
Analyse dynamique .....	59
<i>SSL Pinning Bypass</i> .....	59
<i>Interception du trafic</i> .....	60
<i>Signup</i> .....	60
<i>Création du Master Password</i> .....	62
<i>Se connecter à 1Password</i> .....	63
Conclusion.....	65
<b>Conclusion générale</b> .....	<b>66</b>

## Table des figures :

<b>Figure 1 : Kali Linux</b> .....	<b>8</b>
<b>Figure 2 : Genymotion</b> .....	<b>8</b>
<b>Figure 3 : Apktool</b> .....	<b>9</b>
<b>Figure 4 : Android Debug Bridge</b> .....	<b>9</b>
<b>Figure 5 : Dex2Jar</b> .....	<b>10</b>
<b>Figure 6 : Burpsuite</b> .....	<b>10</b>
<b>Figure 7 : JADX</b> .....	<b>10</b>
<b>Figure 8 : OpenSSL</b> .....	<b>11</b>
<b>Figure 9 : Objection</b> .....	<b>11</b>
<b>Figure 10 : Frida</b> .....	<b>11</b>
<b>Figure 11 : Keeper</b> .....	<b>12</b>
<b>Figure 12: Trusted CA Certificates</b> .....	<b>17</b>
<b>Figure 13: NordPass</b> .....	<b>23</b>
<b>Figure 14 : Différence entre AES et XChaCha20</b> .....	<b>24</b>
<b>Figure 15 : Documentation de java.util.Random</b> .....	<b>27</b>
<b>Figure 16: Frida server (GitHub)</b> .....	<b>28</b>
<b>Figure 17: LastPass</b> .....	<b>33</b>
<b>Figure 18: Bitwarden</b> .....	<b>43</b>
<b>Figure 19: 1Password</b> .....	<b>49</b>
<b>Figure 20: Sign-in details</b> .....	<b>58</b>

## Introduction :

Les mots de passe constituent la première ligne de défense contre l'accès non autorisé à nos informations personnelles. Nous les utilisons pour sécuriser pratiquement tout ce qui est numériquement important pour nous : téléphones mobiles, tablettes, stockage cloud, services bancaires, email, etc ... Le premier conseil de sécurité que l'on puisse donner sur Internet est d'avoir pour chacun de ses comptes en ligne un mot de passe fort, différent des autres. Le second est évidemment d'éviter de garder cette liste de mots de passe sur un post-it sur son écran ou dans un document stocké dans le cloud. La solution la plus efficace et la plus sécurisée est d'utiliser un gestionnaire de mots de passe qui va créer des mots de passe longs et complexes et les rendre accessibles depuis tous vos appareils.

Un gestionnaire de mot de passe est une application qui mémorise les mots de passe et les noms d'utilisateur associés aux différents sites et logiciels que vous utilisez. Tous vos identifiants sont stockés dans une base de données, elle-même chiffrée et accessible par un mot de passe général. Une fois l'application est déverrouillée, elle remplit automatiquement les systèmes d'identification pour un site Internet ou une application sur Smartphone.

Il existe de nombreux gestionnaires de mot de passe sur le marché, mais ils ne sont pas tous égaux. Certains sont plus sécurisés que d'autres.

L'objectif de ce projet est de tester ces applications de gestions de mots de passe en vérifiant si elles gèrent correctement le mot de passe principal ainsi que les clés dérivées associées (KDF, nombre de tours, salts, etc.)

Pour ce faire, nous verrons dans un premier temps comment ces applications sécurisent les données personnelles des utilisateurs en réalisant une analyse statique approfondie, dans un deuxième temps nous effectuerons une analyse dynamique afin de voir quelles données ces applications transfèrent, puis dans un dernier temps, nous évaluerons la fiabilité de ces applications par rapport à leur image marketing.

## Documentation :

Notions techniques :

**Analyse statique** : consiste à analyser le code source d'une application mobile sans exécuter l'application. Elle est généralement utilisée pour détecter des failles de sécurité et d'assurer la mise en œuvre correcte du code. Elle peut être effectuée à l'aide de divers outils tels que des analyseurs de code, des décompilateurs et des debuggers.

**Analyse dynamique** : consiste à examiner le comportement d'une application lors de son exécution, afin de détecter des failles de sécurité et d'autres problèmes qui peuvent être présents dans le code. Ce type d'analyse est très efficace pour identifier des vulnérabilités potentielles, car elle permet de révéler des failles qui ne sont pas forcément apparentes lors d'une analyse statique.

**APK** : Android Package est un format de fichier qu'Android utilise pour distribuer et installer des applications. Par conséquent, un APK contient tous les éléments dont une application a besoin pour s'installer correctement sur un appareil.

## Documentation

Environnement de travail :

**Kali Linux** : est une distribution Linux à code source ouvert, basée sur Debian, destinée aux tests d'intrusion et aux audits de sécurité avancés. Il est conçu pour fournir les outils et les fonctionnalités nécessaires pour accomplir des tâches avancées de test de sécurité et d'intrusion.



Figure 1 : Kali linux <https://www.kali.org/>

**Genymotion** : est un émulateur Android conçu pour les développeurs et testeurs qui permet à ces derniers de tester des applications mobiles sur une variété de dispositifs virtuels avec différentes versions d'Android sans avoir besoin à acheter des dispositifs physiques.



Figure 2 : Genymotion <https://www.genymotion.com/>

## Documentation

Outils utilisés :

**Apktool** : est un outil open-source utilisé principalement pour la rétro-ingénierie (Reverse Engineering) des applications Android. Il permet de décoder les ressources jusqu'à leur forme quasi-originale et de les reconstruire après y avoir effectué des modifications. Il permet également aux analystes d'analyser des applications malveillantes afin d'identifier les vulnérabilités et du code malveillant.



Figure 3: Apktool <https://ibotpeaches.github.io/Apktool/>

**Android Debug Bridge (Adb)** : est un outil qui permet aux développeurs de communiquer avec un appareil (émulateur ou appareil réel) afin de transférer des données entre l'appareil et l'ordinateur, installer des applications ou même exécuter des commandes shell sur l'appareil.



Figure 4 : <https://developer.android.com/studio/command-line/adb>

## Documentation

**Dex2jar** : est un outil principalement utilisé pour convertir les fichiers .dex (*Dalvik Executables*) d'un APK en fichiers .jar. Il fournit également des outils pour décompiler, débuguer et analyser les fichiers Java .jar résultants



Figure 5 : <https://github.com/pxb1988/dex2jar>

**Burpsuite** : est une application Java, développée par *PortSwigger Ltd*, qui peut être utilisée pour la sécurisation ou effectuer des tests d'intrusion sur les applications Web. La suite se compose de différents outils comme un serveur proxy (*Burp Proxy*), robot d'indexation (*Burp Spider*), un outil d'intrusion (*Burp Intruder*), un scanner de vulnérabilités (*Burp Scanner*) et un répéteur HTTP (*Burp Repeater*)



Figure 6 : Burp Suite <https://portswigger.net/burp>

**Jadx** : est un décompilateur open-source qui offre la possibilité de décompiler et d'analyser le code source de toute application Android, ce qui permet de mieux comprendre le fonctionnement de l'application et la façon dont elle a été compilée.



Figure 7 : JADeX <https://github.com/skylot/jadx>

## Documentation

**OpenSSL** : est une bibliothèque de cryptographie qui fournit une implémentation open source des protocoles SSL (Secure Sockets Layer) et TLS (Transport Layer Security). La bibliothèque contient des outils permettant de générer des clés privées RSA, des demandes de signature de certificats (CSR), des sommes de contrôle, de gérer des certificats et d'effectuer des opérations de chiffrement/déchiffrement.



Figure 8: OpenSSL <https://www.openssl.org/>

**Objection** : est un ensemble d'outils d'exploration mobile, optimisé par Frida, et utilisé pour le pentesting Android et iOS. Objection est conçu pour réaliser de nombreuses fonctions comme le contournement de SSL Pinning (SSL Pinning Bypass), le contournement de Root Detection, l'exécution de memory tasks, heap tasks et plus encore sans avoir un dispositif rooté ou jailbroken

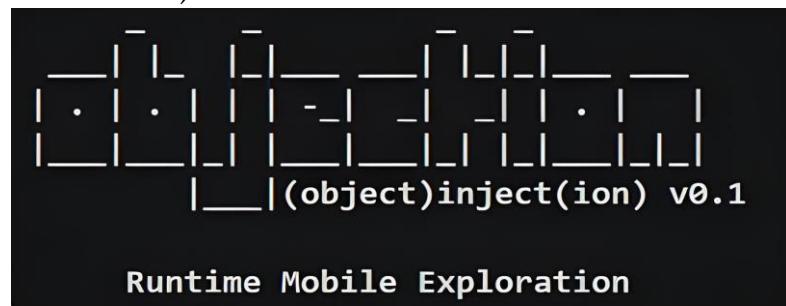


Figure 9 : Objection <https://github.com/sensepost/objection>

**Frida** : est une boîte à outils d'instrumentation dynamique qui permet d'injecter des extraits de code JavaScript dans des applications natives sur Windows, MacOs, GNU/Linux, iOS, Android et QNX. Frida fournit également des outils simples construits au-dessus de l'API Frida. L'un de ces outils est **Objection**.



Figure 10 : Frida <https://frida.re/docs/home/>

## App 1 - Keeper:



Figure 11 : <https://www.keepersecurity.com/>

**Keeper** est l'une des applications les plus prisées du marché des gestionnaires de mots de passe, et cela est justifié. Même si la version gratuite demeure limitée, elle possède des tonnes de fonctionnalités essentielles, comme l'authentification à deux facteurs, le partage sécurisé de mots de passe, ainsi qu'une option de stockage des fichiers et des messages critiques.

### I. Analyse statique :

#### 1. Permissions de l'application :

##### ▪ Permissions dangereuses :

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

Cette permission permet à l'application de prendre des photos et des vidéos et de collecter les images que la caméra peut voir à tout moment.

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

Cette permission permet à l'application d'accéder à la liste des comptes du *Accounts Service*

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

Cette permission permet à l'application de lire tous les contacts et adresses stockés sur le téléphone.

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

Cette permission permet à l'application de lire à partir d'un stockage externe.

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

Cette permission permet à l'application d'accéder aux enregistrement audio.

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

Cette permission permet à l'application de lire/modifier/supprimer le contenu du stockage externe.

## App1 - Keeper

### 2. Protection des données :

Keeper est un fournisseur de sécurité qui suit le modèle ‘Zero-Knowledge’. En effet, L’utilisateur Keeper est la seule personne à avoir un contrôle total sur le cryptage et le décryptage de ses données.

Chaque dossier stocké dans le coffre-fort (Vault) est crypté avec une clé **AES-256 bits** généré aléatoirement sur l’appareil de l’utilisateur

```
private static void generateKey(KeyGenParameterSpec keyGenParameterSpec) throws GeneralSecurityException {
    KeyGenerator keyGenerator = KeyGenerator.getInstance("AES", ANDROID_KEYSTORE);
    keyGenerator.init(keyGenParameterSpec);
    keyGenerator.generateKey();
}
```

### 3. PBKDF2 :

La clé de données (DataKey) est chiffrée par une clé dérivée du mot de Passe principal (MasterKey) sur l’appareil de l’utilisateur.

Cette MasterKey est dérivée en utilisant **PBKDF2** (*Password Based Key Derivation Function*) avec 100,000 itérations.

```
public class PBKDF2Key implements PBKDFKey {
    private final CharToByteConverter converter;
    private final char[] password;

    @Override // java.security.Key
    public String getAlgorithm() {
        return "PBKDF2";
    }

    public PBKDF2Key(char[] cArr, CharToByteConverter charToByteConverter) {
        this.password = Arrays.clone(cArr);
        this.converter = charToByteConverter;
    }

    public char[] getPassword() {
        return this.password;
    }

    public static final int KEY_SIZE_IN_BITS = 256;
    private static final int MINIMUM_ITERATIONS = 100;
    private static final int PBKDF2_RESULT_LENGTH = 32;
    private static final int PBKDF2_RESULT_OFFSET = 20;
    private static final int SALT_LENGTH = 16;
    private static final int SALT_OFFSET = 4;
```

### 4. Chiffrement AES-256-GCM :

Keepass utilise le chiffrement par bloc **AES-256-GCM** (*Advanced Encryption Standard in Galois Counter Mode*), qui offre une vitesse élevée de chiffrement d’authenticité et d’intégrité des données.

```
public final class MasterKeys {
    private static final String ANDROID_KEYSTORE = "AndroidKeyStore";
    static final String KEYSTORE_PATH_URI = "android-keystore://";
    private static final int KEY_SIZE = 256;
    static final String MASTER_KEY_ALIAS = "_androidx_security_master_key_";
    public static final KeyGenParameterSpec AES256_GCM_SPEC = createAES256GCMKeyGenParameterSpec(MASTER_KEY_ALIAS);
```

### 5. Chiffrement client :

Les données stockées sur l’appareil de l’utilisateur sont également cryptées

## App1 - Keeper

par une autre clé AES-256 bits, appelée ClientKey.

```
public class ClientKey {
    private static final int CLIENT_KEY_OFFSET = 19;
    private static final int ITERATION_COUNT_LENGTH = 3;
    private static final int ITERATION_COUNT_OFFSET = 0;
    private static final int SALT_LENGTH = 16;
    private static final int SALT_OFFSET = 3;
    private byte[] encryptedClientKey;
    private int iterations;
    private byte[] salt;

    public static ClientKey createAndSave(String password, final byte[] decryptedClientKey, LocalKeyType keyType) {
        return new LocalKeyProcessor().createAndSaveClientKey(decryptedClientKey, password, keyType);
    }

    public static ClientKey create(String password, final byte[] decryptedClientKey) {
        return new LocalKeyProcessor().createClientKey(decryptedClientKey, password);
    }

    public ClientKey(byte[] encodedBytes) throws IllegalArgumentException {
        if (encodedBytes == null) {
            throw new IllegalArgumentException("input must not be null");
        }
        try {
            this.iterations = decodeIterations(encodedBytes);
            this.salt = subArray(encodedBytes, 3, 16);
            this.encryptedClientKey = subArray(encodedBytes, 19, encodedBytes.length - 19);
            verify();
        } catch (ArrayIndexOutOfBoundsException e) {
            throw new IllegalArgumentException(e);
        }
    }

    public ClientKey(int iterations, byte[] salt, byte[] encryptedClientKey) throws IllegalArgumentException {
        this.salt = salt;
        this.iterations = iterations;
        this.encryptedClientKey = encryptedClientKey;
        verify();
    }
}
```

## 6. Record-level Encryption :

Keeper implémente un système de chiffrement à plusieurs niveaux (*multi-layered encryption system*) base sur des clés générées par le client. Les clés au niveau de l'archive (Record-level keys) et au niveau du dossier (Folder-level keys) sont générées sur l'appareil local et permettent de chiffrer chaque archive du coffre.

Par exemple, si vous avez 10.000 archives dans votre coffre-fort, vous avez également 10.000 clés AES d'archives qui protègent les données.

```
protected KeyParameter generateRecordMACKey(StreamCipher streamCipher) {
    byte[] bArr = new byte[64];
    streamCipher.processBytes(bArr, 0, 64, bArr, 0);
    KeyParameter keyParameter = new KeyParameter(bArr, 0, 32);
    Arrays.fill(bArr, (byte) 0);
    return keyParameter;
}

public final byte[] encryptRecordKey(byte[] recordKey, byte[] recipientsPublicKey) {
    Intrinsics.checkNotNullParameter(recordKey, "recordKey");
    Intrinsics.checkNotNullParameter(recipientsPublicKey, "recipientsPublicKey");
    EcCryptoHelper ecCryptoHelper = new EcCryptoHelper();
    EcKeyPairRaw createRawEcKeyPair = ecCryptoHelper.createRawEcKeyPair();
    return concatenate(createRawEcKeyPair.getPublicKeyBytes(), gcmEncrypt(sha256(generateSharedSecret(
        recordKey, recipientsPublicKey, createRawEcKeyPair.getPublicKeyBytes())
    ), createRawEcKeyPair.getPrivateKeyBytes()));
}
```

## 7. Two-Factor Authentication :

Pour se protéger contre tout accès non autorisé au compte d'un client, Keeper propose également l'authentification à deux facteurs (Two-Factor Authentication), qui est une méthode d'authentification qui nécessite au moins deux des trois facteurs d'authentification : un facteur de connaissance, un facteur de possession et un facteur d'héritage.

Pour ce faire, Keeper utilise quelque chose que vous connaissez (votre mot de passe) et quelque chose que vous avez (numéro de téléphone, adresse

## App1 - Keeper

email etc...)

```
public enum TwoFactorAuthenticationType {
    Email("two_factor_channel_email"),
    Google("two_factor_channel_google"),
    RSA("two_factor_channel_rsa"),
    SMS("two_factor_channel_sms"),
    Voice("two_factor_channel_voice"),
    Push("two_factor_channel_push"),
    Duo("two_factor_channel_duo"),
    Disabled("two_factor_disabled"),
    NONE("");
}

private final String channel;

TwoFactorAuthenticationType(String channel) {
    this.channel = channel;
}

public static TwoFactorAuthenticationType getAuthenticationType(String userChannel) {
    TwoFactorAuthenticationType[] values;
    if (userChannel == null || Disabled.channel.equals(userChannel)) {
        return NONE;
    }
    for (TwoFactorAuthenticationType twoFactorAuthenticationType : values()) {
        if (twoFactorAuthenticationType.getChannel().equals(userChannel)) {
            return twoFactorAuthenticationType;
        }
    }
    return NONE;
}
```

Pour cela, Keeper génère une clé secrète de 10 octets à l'aide d'un générateur pseudo aléatoire cryptographique sûr. Ce code est valable pendant environ 10 minutes et est envoyé à l'utilisateur par SMS, Duo Security, RSA SecurID, une application TOTP, ou Google Authenticator



SMS



Duo Security



RSA SecurID



Google Authenticator



Microsoft Authenticator

### 8. Algorithme TOTP :

Keeper utilise l'algorithme TOTP (Time-based One-Time Password) pour implémenter le Two-Factor Authentication :

```
public class TimeBasedOneTimePasswordGenerator extends HmacOneTimePasswordGenerator {
    public static final Duration DEFAULT_TIME_STEP = Duration.ofSeconds(30);
    public static final String TOTP_ALGORITHM_HMAC_SHA1 = "HmacSHA1";
    public static final String TOTP_ALGORITHM_HMAC_SHA256 = "HmacSHA256";
    public static final String TOTP_ALGORITHM_HMAC_SHA512 = "HmacSHA512";
    private final Duration timeStep;

    public TimeBasedOneTimePasswordGenerator() throws NoSuchAlgorithmException {
        this(DEFAULT_TIME_STEP);
    }

    public TimeBasedOneTimePasswordGenerator(final Duration timeStep) throws NoSuchAlgorithmException {
        this(timeStep, 6);
    }

    public TimeBasedOneTimePasswordGenerator(final Duration timeStep, final int passwordLength) throws NoSuchAlgorithmException {
        this(timeStep, passwordLength, "HmacSHA1");
    }

    public TimeBasedOneTimePasswordGenerator(final Duration timeStep, final int passwordLength, final String algorithm) throws NoSuchAlgorithmException {
        super(passwordLength, algorithm);
        this.timeStep = timeStep;
    }

    public int generateOneTimePassword(final Key key, final Instant timestamp) throws InvalidKeyException {
        return generateOneTimePassword(key, timestamp.toEpochMilli() / this.timeStep.toMillis());
    }

    public Duration getTimeStep() {
        return this.timeStep;
    }
}
```

## App1 - Keeper

### 9. Identification biométrique :

L'identification biométrique est utilisée pour vérifier l'identité d'une personne en se basant sur ses caractéristiques physiques (empreintes digitales, reconnaissance facial, reconnaissance vocale etc ...)

Keeper prend en charge les systèmes biométriques : Windows Hello, Touch ID, Face ID et Android biometrics

Si la biométrie est activée, une clé biométrique AES-256 bits est générée localement sur l'appareil, puis stockée dans le trousseau iOS (iOS keychain)

```
private final byte[] generateBiometricKey() {
    SecretKey generate256bitAESSecretKey = EncryptionUtil.generate256bitAESSecretKey();
    byte[] encoded = generate256bitAESSecretKey == null ? null : generate256bitAESSecretKey.getEncoded();
    if (encoded == null) {
        return null;
    }
    return encoded;
}
```

Quand l'utilisateur s'authentifie avec succès auprès du système d'exploitation en utilisant l'authentification Face ID ou Touch ID, la clé est fourni à l'application Keeper pour révéler (unwrap) le texte chiffré stocké et éventuellement récupérer la clé de données.

### 10. Vérification du certificat du serveur WebView :

Implémentation non sécurisée de WebView. En effet, WebView ignore les Erreurs de certificat SSL et accepte n'importe quel certificat SSL.

Cette application est vulnérable aux attaques MITM (*Man In The Middle*)

```
private void setupWebView() {
    setupWebViewSettings();
    this.keeperFill = new KeeperFill(this.webView, this.keeperFillListener);
    this.webView.setWebChromeClient(this.webChromeClient);
    this.webView.requestFocus(130);
    this.webView.setWebViewClient(this.webViewClient);
    this.webView.setOnTouchListener(WebActivity$$ExternalSyntheticLambda18.INSTANCE);
    this.webView.setDownloadListener(new DownloadListener() { // from class: com.callpod.android_apps.
        {
            WebActivity.this = this;
        }

        @Override // android.webkit.DownloadListener
        public final void onDownloadStart(String str, String str2, String str3, String str4, long j) {
            WebActivity.this.m4449x6a91ba23(str, str2, str3, str4, j);
        }
    });
}
```

### 11. PKCS5/PKCS7 :

L'application utilise le mode de chiffrement CBC (Cipher Block Chaining) Avec le padding PKCS5/PKCS7.

Cette configuration est vulnérable aux attaques (*Padding Oracle Attacks*)

<https://eklitzke.org/the-cbc-padding-oracle-problem>

```
public class PKCS7Padding implements BlockCipherPadding {
    @Override // org.bouncycastle.crypto.paddings.BlockCipherPadding
    public int addPadding(byte[] bArr, int i) {
        byte length = (byte) (bArr.length - i);
        while (i < bArr.length) {
            bArr[i] = length;
            i++;
        }
        return length;
    }

    @Override // org.bouncycastle.crypto.paddings.BlockCipherPadding
    public String getPaddingName() {
        return "PKCS7";
    }
}
```

### II. Analyse dynamique :

#### 1. Configuration de Burp Suite :

A partir de Nougat (Android 7.0), Android a modifié le comportement par défaut qui consiste à faire confiance aux certificats installés par l'utilisateur. Il n'est plus possible de simplement installer l'autorité de certification Burp CA depuis la sdcard afin de pouvoir intercepter le trafic des applications.

Sauf indication contraire, les applications ne feront désormais confiance qu'aux CA du système situé sous le chemin `/system/etc/security/cacerts/`.

← Trusted credentials	
SYSTEM	USER
AC Camerfirma S.A. Chambers of Commerce Root - 2008	<input checked="" type="checkbox"/>
AC Camerfirma S.A. Global Chambersign Root - 2008	<input checked="" type="checkbox"/>
ACCV ACCVRAIZ1	<input checked="" type="checkbox"/>
Actalis S.p.A./03358520967 Actalis Authentication Root CA	<input checked="" type="checkbox"/>
AddTrust AB AddTrust External CA Root	<input checked="" type="checkbox"/>
AffirmTrust AffirmTrust Commercial	<input checked="" type="checkbox"/>

Figure 12 : Trusted CA certificates

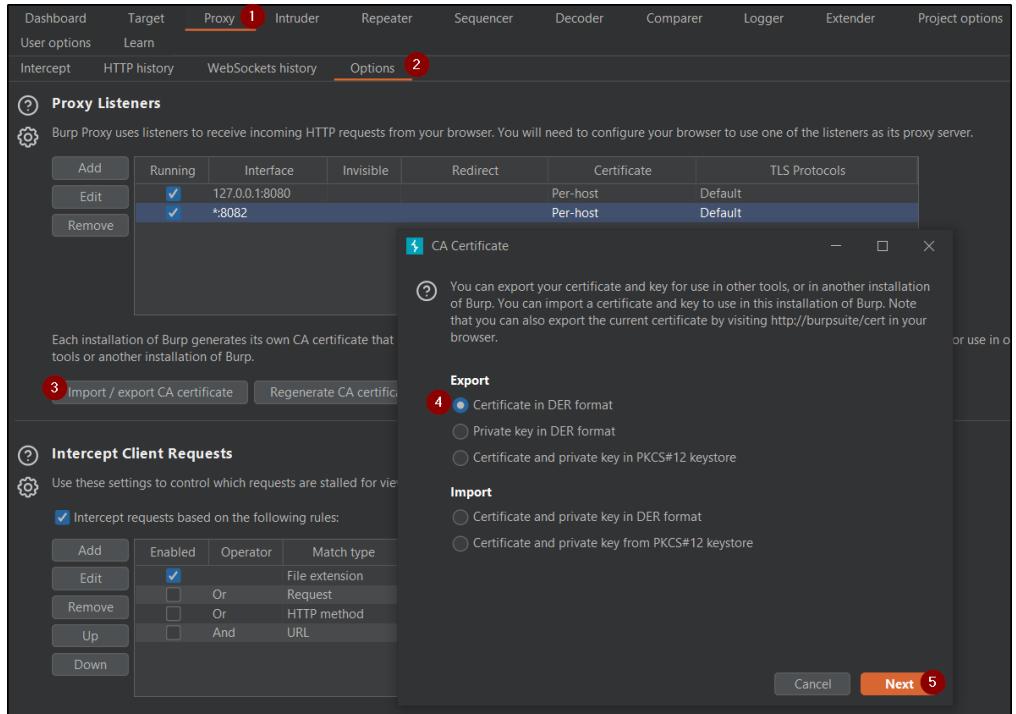
Pour contourner ce mécanisme, il faut installer manuellement Burp CA dans les CA du système.

Trusted CAs pour Android sont stockées dans un format spécifique dans `/system/etc/security/cacerts`. Si nous disposons des privilège root, il est possible d'écrire à cet emplacement et d'insérer le Burp CA.

#### ✓ Installation de Burp CA :

La première étape consiste à exporter le CA de Burp Suite au format DER.

## App1 - Keeper



### ✓ Renommer le CA Certificate :

Trusted CAs pour Android sont stockées dans un format spécifique dans `/system/etc/security/cacerts`

```
vbox86p:/ # cd /system/etc/security/cacerts
vbox86p:/system/etc/security/cacerts # ls -l
total 1040
-rw-r--r-- 1 root root      4767 2022-10-18 10:47 00673b5b.0
-rw-r--r-- 1 root root      2910 2022-10-18 10:47 04f60c28.0
-rw-r--r-- 1 root root     2341 2022-10-18 10:47 0d69c7e1.0
-rw-r--r-- 1 root root     4614 2022-10-18 10:47 10531352.0
-rw-r--r-- 1 root root     4716 2022-10-18 10:47 111e6273.0
-rw-r--r-- 1 root root     4300 2022-10-18 10:47 12d55845.0
-rw-r--r-- 1 root root     4578 2022-10-18 10:47 17b51fe6.0
-rw-r--r-- 1 root root     7088 2022-10-18 10:47 1dcfd6f4c.0
-rw-r--r-- 1 root root     4923 2022-10-18 10:47 1df5a75f.0
-rw-r--r-- 1 root root     4527 2022-10-18 10:47 1e1eab7c.0
-rw-r--r-- 1 root root     4332 2022-10-18 10:47 1e8e7201.0
-rw-r--r-- 1 root root     8399 2022-10-18 10:47 1eb37bdf.0
-rw-r--r-- 1 root root     7076 2022-10-18 10:47 1f58a078.0
-rw-r--r-- 1 root root     7054 2022-10-18 10:47 21855f49.0
```

Android exige que le certificat soit au format PEM et que le nom du fichier soit `subject_hash_old` précédé de .o (zéro)

On va utiliser `openssl` pour convertir DER en PEM, puis générer le `subject_hash_old` et renommer le fichier qu'on a exporté de Burp.

```
[(kali㉿kali)-~]
$ openssl x509 -inform DER -in cert.der -out cacert.pem

[(kali㉿kali)-~]
$ openssl x509 -inform PEM -subject_hash_old -in cacert.pem |head -1
9a5ba575

[(kali㉿kali)-~]
$ mv cacert.pem 9a5ba575.0
```

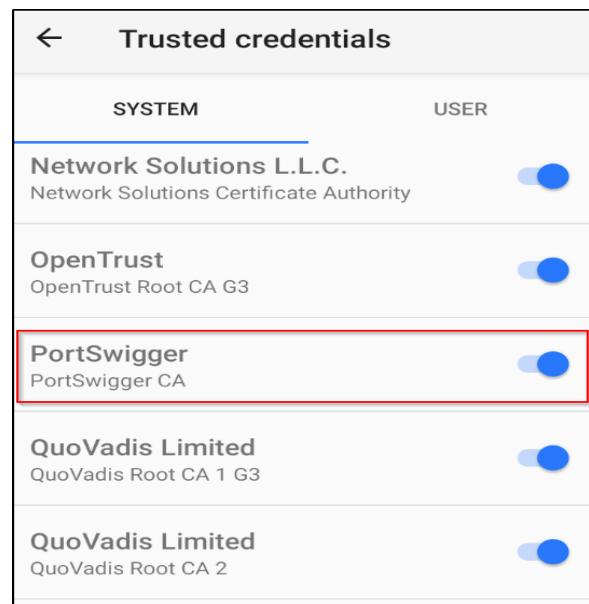
## App1 - Keeper

### ✓ Copier le certificat sur l'appareil :

Nous pouvons utiliser **adb** pour copier le certificat, mais il faut d'abord remonter la partition **/system** pour qu'on puisse écrire et ajouter le certificat sous le chemin **/system/etc/security/cacerts**

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb.exe root  
adb is already running as root  
  
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb remount  
remount succeeded  
  
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb push C:\Users\pc\Desktop\9a5ba575.0 /sdcard/  
C:\Users\pc\Desktop\9a5ba575.0: 1 file pushed, 0 skipped. 0.1 MB/s (1326 bytes in 0.010s)  
  
vbox86p:/ # mv /sdcard/9a5ba575.0 /system/etc/security/cacerts  
vbox86p:/ # chmod 644 /system/etc/security/cacerts/9  
91739615.0 9339512a.0 9576d26b.0 9685a493.0 985c1f52.0 9d6523ce.0  
9282e51c.0 9479c8c3.0 95aff9e3.0 9772ca32.0 9a5ba575.0 9f533518.0  
vbox86p:/ # chmod 644 /system/etc/security/cacerts/9a5ba575.0  
vbox86p:/ # reboot
```

Après le redémarrage de l'appareil, la navigation dans *Settings -> Security -> Trusted Credentials* devrait montrer l'autorité de certification '**Portswigger CA**' comme *system trusted CA*



➔ Il est maintenant possible de configurer le proxy Burp et de commencer l'interception du trafic issue de notre application Keeper.

## App1 - Keeper

### 2. Interception du trafic :

#### ▪ Signup :

La saisie de l'adresse email :

The figure consists of three vertically stacked screenshots from a mobile application. Each screenshot shows a 'Create Account' screen with a 'NEXT' button and a 'Already have an account? Login' link. On the left, there is a NetworkMiner-like interface showing a POST request to '/api/rest/authentication/get\_domain\_password\_rules'. The third screenshot shows a second POST request to '/api/rest/authentication/start\_login'. Each screenshot has a red box highlighting the raw POST data sent by the app.

**Screenshot 1 (Top):** The raw POST data is:

```
POST /api/rest/authentication/get_domain_password_rules HTTP/2
Host: keepersecurity.com
Content-Type: application/octet-stream
Content-Length: 324
Accept-Encoding: gzip, deflate
User-Agent: Callpod Keeper for Android 1.0 (16.6.10.1075/1075) Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQID.200105.002)
```

**Screenshot 2 (Middle):** The raw POST data is:

```
POST /api/rest/authentication/get_domain_password_rules HTTP/2
Host: keepersecurity.com
Content-Type: application/octet-stream
Content-Length: 324
Accept-Encoding: gzip, deflate
User-Agent: Callpod Keeper for Android 1.0 (16.6.10.1075/1075) Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQID.200105.002)
```

**Screenshot 3 (Bottom):** The raw POST data is:

```
POST /api/rest/authentication/start_login HTTP/2
Host: keepersecurity.com
Content-Type: application/octet-stream
Content-Length: 374
Accept-Encoding: gzip, deflate
User-Agent: Callpod Keeper for Android 1.0 (16.6.10.1075/1075) Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQID.200105.002)
```

## App1 - Keeper

#### La saisie du Master Password :

#### **La saisie du code de vérification :**

<pre>POST /api/test/authentication/validate create user verification code HTTP/2 Host : keepersecurity.com Content-Type : application/octet-stream Content-Length : 346 Accept-Encoding : gzip, deflate User-Agent : Callpod Keeper for Android 1.0 (16.6.10.1075/1075) Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQD.200105.002)</pre>	<p>An email has been sent to tasrayounes36.yt@gmail.com with instructions. If a verification code is provided, you may enter it below.</p> <p>Verification Code</p> <p>HX4E6V</p> <p>Resend Code</p>
<pre>POST /api/test/login/account_summary HTTP/2 Host : keepersecurity.com Content-Type : application/octet-stream Content-Length : 366 Accept-Encoding : gzip, deflate User-Agent : Callpod Keeper for Android 1.0 (16.6.10.1075/1075) Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQD.200105.002)</pre>	<p>An email has been sent to tasrayounes36.yt@gmail.com with instructions. If a verification code is provided, you may enter it below.</p> <p>Verification Code</p> <p>HX4E6V</p> <p>Resend Code</p>
<pre>POST /emergency_check HTTP/2 Host : keepersecurity.com Content-Type : application/x-www-form-urlencoded Content-Length : 407 Accept-Encoding : gzip, deflate User-Agent : Callpod Keeper for Android 1.0 (16.6.10.1075/1075) Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQD.200105.002)</pre>	<p>An email has been sent to tasrayounes36.yt@gmail.com with instructions. If a verification code is provided, you may enter it below.</p> <p>Verification Code</p> <p>HX4E6V</p> <p>Resend Code</p>
<pre>country=US&amp;product=vbox86p&amp;code=&amp;appstore_type=&amp;mcn_mnc=310270&amp;appstore_token=&amp;format=json&amp;pid=441&amp;language=en&amp;mfy=Genymobile&amp;session_token= 5ikHvbxzNrk2fzmG01JN4CstMp5KpQ17en1zcVjMPhRmno_GTuDzqVRfex0bhaxJhDtwrWhPtpI204 &amp;uid= 3Ww0UB_Ford_sHeogPfQ &amp;carrier=Android&amp;sv=16.6.10 &amp;rcrs=0&amp;device=vbox86p &amp;theme=CLASSIC_BLUE &amp;model=Nexus+4 &amp;brand=Google &amp;build_type=gplayProduction &amp;email=tasrayounes36.yt@gmail.com</pre>	<p>An email has been sent to tasrayounes36.yt@gmail.com with instructions. If a verification code is provided, you may enter it below.</p> <p>Verification Code</p> <p>HX4E6V</p> <p>Resend Code</p>

- ➔ Il semble que le trafic dirigeant vers **/api/rest/authentication/** est chiffré, ce qui signifie qu'un attaquant ne pourra jamais voir l'adresse email et le Master Password en clair même s'il parvient à intercepter le trafic.

## App1 - Keeper

➔ Par contre, le trafic allant vers **/emergency\_check** n'est pas chiffré et nous pouvons clairement lire l'adresse email de l'utilisateur et quelques autres données envoyés par l'application.

### ▪ Vault Login :

La saisie de l'adresse email :

The figure consists of three vertically stacked screenshots from NetworkMiner, a network traffic analysis tool. Each screenshot shows a different stage of the login process:

- Screenshot 1 (Top):** Shows the initial POST request to `/api/rest/authentication/start_login`. The "Email" field contains `tasrayounes36.yt@gmail.com`. The raw data shows the email address being sent in plain text.
- Screenshot 2 (Middle):** Shows the POST request to `/api/rest/authentication/validate_auth_hash`. The "Master Password" field is filled with a series of dots, and the raw data shows the password being sent in plain text.
- Screenshot 3 (Bottom):** Shows the final POST request to `/api/rest/login/account_summary`. The raw data shows the user account information being sent in plain text.

Each screenshot also includes a "Vault Login" interface on the right side of the NetworkMiner window, which displays the entered email and password fields.

### ▪ Security Question :

A screenshot of the "Security Question" screen in the Keeper mobile application. The screen has a dark theme with yellow accents. It includes fields for "Security Question" (with placeholder "who is your favourite football player") and "Security Answer" (with placeholder "Lionel Andrés Messi"). Below these fields is a "Confirm Security Answer" button and a text input field containing "Lionel Andrés Messi". At the bottom right is a large yellow circular progress bar.

## IV. Conclusion :

Properties	
Dangerous permissions	7
Encryption Standard	AES-256-GCM
Key Expansion Algorithm	PBKDF <sub>2</sub>
Padding	PKCS <sub>5</sub> /PKCS <sub>7</sub>
Vulnerable to padding Oracle Attacks	✓
Secure PRG	✗
Secure Hashing Algorithms	✗
Secure PRG	✓
Zero Knowledge Architecture	✓
2FA Authentication	✓
MFA Authentication	✓
Biometrics	✓
Security alerts	✓
Mobile application	✓
Security audits	✓
Import of passwords	✓
Export of passwords	✓
SSO Authentication	✓
Password sharing	✓
Integrated database	✓
Certificate Pinning	✗
Root detection	✓
Encrypted traffic	✓
Encrypted email	✓
Encrypted Master Password	✓

## App 2 - NordPass :



Figure 13 : <https://nordpass.com/homepage/>

**NordPass** est un gestionnaire de mot de passe conçu par les experts en Cybersécurité de NordVPN. **NordPass** utilise l'architecture *zero-knowledge* pour assurer la sécurité de vos informations d'identification importantes.

**NordPass** permet de sauvegarder automatiquement tous vos mots de passe, remplir automatiquement des formulaires en ligne, sauvegarder en toute sécurité vos notes privées, et générer des mots de passe robustes et aléatoires.

### I. Analyse statique :

#### 1. Permissions de l'application :

##### ▪ Permissions dangereuses :

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

Cette permission permet à l'application de prendre des photos et des vidéos et de collecter les images que la caméra peut voir à tout moment.

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

Cette permission permet à l'application de lire tous les contacts et adresses stockés sur le téléphone.

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

Cette permission permet à l'application de lire/modifier/supprimer le contenu du stockage externe.

##### ▪ Permissions inconnues :

```
<uses-permission android:name="com.android.vending.BILLING"/>
```

```
<uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
```

```
<uses-permission android:name="com.nordpass.android.app.password.manager.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/>
```

##### ▪ Permissions normales :

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

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

Ces permissions permettent à l'application d'utiliser les modalités biométriques supportés par l'appareil

## App2 - NordPass

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

Ces permissions permettent à l'application de :

- ➔ Afficher des informations sur l'état du Wi-Fi.
- ➔ Visualiser l'état du réseau.
- ➔ Créer des sockets réseau.

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

Cette permission permet à l'application de communiquer avec des étiquettes, cartes et des lecteurs NFC (*Near-Field Communication*)

### 2. Protection des données :

**NordPass** est un fournisseur de sécurité qui suit le modèle 'Zero-Knowledge'. En effet, le chiffrement se déroule sur les ordinateurs ou les smartphones avant d'être envoyé sur les serveurs du logiciel.

Autrement dit, personne, y compris l'équipe NordPass, ne peut voir les mots de passe stockés dans l'application.

NordPass utilise l'algorithme de chiffrement **XChaCha20** qui est un chiffrement par flux de 256 bits (Stream Cipher). Comme AES, XChaCh20 est symétrique et utilise une seule clé pour chiffrer et déchiffrer les données (Il existe également une version asymétrique)

```
public native int crypto_aead_xchacha20poly1305_ietf_decrypt(byte[] bArr, long[] jArr, byte[] bArr2, byte[] bArr3, long j, byte[] bArr4, long j2, byte[] bArr5, byte[] bArr6);
public native int crypto_aead_xchacha20poly1305_ietf_encrypt(byte[] bArr, long[] jArr, byte[] bArr2, long j, byte[] bArr3, long j2, byte[] bArr4, byte[] bArr5, byte[] bArr6);
```

Le chiffrement par flux à 20 tours ChaCha20 est toujours plus rapide et moins sensible aux Timing Attacks que l'algorithme AES, puisqu'il met en œuvre deux longueurs de clés différentes, dont le chiffrement de 256 bits.

AES	XChaCha20
1. 128, 192 or 256 bits	256 bits
2. Block cipher	Stream cipher
3. Old	New
4. Complex	Simpler
5. Prone to human error	Not as prone to human error
6. Requires hardware	Does not require hardware
7. Can be slow	Faster

Figure 14 : Différence entre AES et XChaCha20

### 3. Utilisation du presse-papiers (Clipboard) :

Lorsque des données sont saisies, le presse-papiers peut être utilisé pour copier des données. Le presse-papiers est accessible à l'ensemble du système et est donc partagé par les applications. Ce partage peut être utilisé par des applications malveillantes pour obtenir des données sensibles qui enregistrées dans le presse-papiers.

## App2 - NordPass

L'application **NordPass** copie des données dans le presse-papiers, ce qui signifie qu'une application malveillante peut avoir accès à ces données.

```
public final void a(String str) {
    ClipboardManager clipboardManager;
    uc0.l.f(str, "text");
    ClipData newPlainText = ClipData.newPlainText(str, str);
    ClipDescription description = newPlainText.getDescription();
    PersistableBundle persistableBundle = new PersistableBundle();
    persistableBundle.putBoolean("android.content.extra.IS_SENSITIVE", true);
    description.setExtras(persistableBundle);
    try {
        Object systemService = this.f24916a.getSystemService("clipboard");
        uc0.l.d(systemService, "null cannot be cast to non-null type android.content.ClipboardManager");
        clipboardManager = (ClipboardManager) systemService;
    } catch (Throwable th2) {
        this.f24917b.a(th2);
        clipboardManager = null;
    }
    if (clipboardManager != null) {
        clipboardManager.setPrimaryClip(newPlainText);
    }
}
```

### 4. Certificate Pinning :

*Certificate Pinning* est le processus qui associe le serveur backend avec un certificat particulier **X.509** ou une clé publique spécifique au lieu d'accepter n'importe quel certificat signé par une autorité de certification. Après avoir stocké (*Pinning*) le certificat ou la clé publique, l'application mobile se connectera uniquement au serveur connu.

L'application **NordPass** utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques *MITM* (*Man In The Middle*) dans des canaux de communication sécurisés.

```
public SSLSocketFactory m(X509TrustManager x509TrustManager) {
    try {
        SSLContext l11 = l();
        l11.init(null, new TrustManager[]{x509TrustManager}, null);
        SSLSocketFactory socketFactory = l11.getSocketFactory();
        l.e(socketFactory, "newSSLSocket().apply {\n...l11\n    }.socketFactory");
        return socketFactory;
    } catch (GeneralSecurityException e11) {
        throw new AssertionError(l.k(e11, "No System TLS: "), e11);
    }
}

public X509TrustManager n() {
    TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
    trustManagerFactory.init((KeyStore) null);
    TrustManager[] trustManagers = trustManagerFactory.getTrustManagers();
    l.c(trustManagers);
    boolean z4 = true;
    if (trustManagers.length != 1 || !(trustManagers[0] instanceof X509TrustManager)) {
        z4 = false;
    }
    if (!z4) {
        String arrays = Arrays.toString(trustManagers);
        l.e(arrays, "toString(this)");
        throw new IllegalStateException(l.k(arrays, "Unexpected default trust managers: ").toString());
    }
    TrustManager trustManager = trustManagers[0];
    if (trustManager != null) {
        return (X509TrustManager) trustManager;
    }
    throw new NullPointerException("null cannot be cast to non-null type javax.net.ssl.X509TrustManager");
}
```

### 5. Stockage externe :

Tous les appareils Android prennent en charge le stockage externe partagé. Ce stockage peut être amovible (Carte SD) ou interne. Les

## App2 - NordPass

fichiers enregistrés sur le stockage externe sont accessibles par une partie tierce.

L'application **NordPass** a les permissions de lecture/écriture sur le stockage externe, ce qui signifie que toute application malveillante installée sur le smartphone peut lire les données enregistrées sur le stockage externe.

```
public static String getPath(Context context, Uri uri) {
    Uri uri2 = null;
    if (DocumentsContract.isDocumentUri(context, uri)) {
        if (isExternalStorageDocument(uri)) {
            String[] split = DocumentsContract.getDocumentId(uri).split(":");
            if ("primary".equalsIgnoreCase(split[0])) {
                return Environment.getExternalStorageDirectory() + "/" + split[1];
            }
        }
    }
    private boolean isExternalStorageMounted() {
        String externalStorageState = Environment.getExternalStorageState();
        return ("mounted".equals(externalStorageState) || "mounted_ro".equals(externalStorageState)) && !Environment.isExternalStorageEmulated();
    }
}
```

### 6. Root detection :

L'objectif de *Root Detection* est de rendre l'exécution d'une application sur un appareil rooté un peu plus difficile, ce qui empêche l'utilisation des outils et techniques de *reverse engineering*.

L'application **NordPass** implémente le *Root Detection*

```
public Map<String, Object> lambda$new$0() {
    HashMap hashMap = new HashMap();
    hashMap.put(ROOTED, Boolean.valueOf(this.rootChecker.isDeviceRooted()));
    String kernelVersion = getKernelVersion();
    if (kernelVersion != null) {
        hashMap.put(KERNEL_VERSION, kernelVersion);
    }
    hashMap.put(EMULATOR, isEmulator());
    Map<String, String> sideLoadedInfo = getSideLoadedInfo();
    if (sideLoadedInfo != null) {
        hashMap.put(SIDE_LOADED, sideLoadedInfo);
    }
    return hashMap;
}

public final class RootChecker {
    private static final Charset UTF_8 = Charset.forName("UTF-8");
    private final IBuildInfoProvider buildInfoProvider;
    private final Context context;
    private final ILogger logger;
    private final String[] rootFiles;
    private final String[] rootPackages;
    private final Runtime runtime;

    public RootChecker(Context context, IBuildInfoProvider iBuildInfoProvider, ILogger iLogger) {
        this(context, iBuildInfoProvider, iLogger, new String[]{"system/app/Superuser.apk", "/sbin/su", "/system/bin/su", "/system/xbin/su"});
    }
}
```

### 7. Algorithmes cryptographiques non sécurisés :

L'application NordPass utilise un algorithme cryptographique cassé et non sécurisé à savoir MD5, qui entraîne des collisions de hachage.

Ces collisions signifient qu'un attaquant pourrait créer des fichiers qui auraient exactement le même hachage qu'un autre, ce qui rend impossible que le fichier n'a pas été falsifié.

## App2 - NordPass

```
final class WebSocketUtil {
    public static final /* synthetic */ boolean $assertionsDisabled = false;
    private static final FastThreadLocal<MessageDigest> MD5 = new FastThreadLocal<MessageDigest>() {
        @Override // io.netty.util.concurrent.FastThreadLocal
        public MessageDigest initialValue() {
            try {
                return MessageDigest.getInstance("MD5");
            } catch (NoSuchAlgorithmException unused) {
                throw new InternalError("MD5 not supported on this platform - Outdated?");
            }
        }
    }

    public static MessageDigest m4301D() {
        MessageDigest messageDigest;
        for (int i = 0; i < 2; i++) {
            try {
                messageDigest = MessageDigest.getInstance("MD5");
            } catch (NoSuchAlgorithmException unused) {
            }
            if (messageDigest != null) {
                return messageDigest;
            }
        }
        return null;
    }
}
```

### 8. Générateurs de nombres aléatoires non sécurisés :

Il est fondamentalement impossible de générer des nombres strictement aléatoires sur un dispositif déterministe. Les générateurs pseudo-aléatoires compensent ce problème en produisant un flux de nombre pseudo-aléatoires, c'est-à-dire un flux de nombres qui semblent avoir été générés aléatoirement.

L'application **NordPass** utilise un générateur de nombres aléatoires non sécurisé `java.util.Random`

```
import java.util.Random;
import uc0.C182731;

/* renamed from: xc0.b */
/* Loaded from: classes2.dex */
public final class C20015b extends AbstractC20014a {

    /* renamed from: w */
    public final C20016a f46674w = new C20016a();

    /* renamed from: xc0.b$a */
    /* Loaded from: classes2.dex */
    public static final class C20016a extends ThreadLocal<Random> {
        @Override // java.lang.ThreadLocal
        public final Random initialValue() {
            return new Random();
        }
    }

    @Override // xc0.AbstractC20014a
    /* renamed from: h */
    public final Random mo481h() {
        Random random = this.f46674w.get();
        C182731.m2054e(random, "implStorage.get()");
        return random;
    }
}
```

La documentation de `java.util.Random` énonce explicitement ce qui suit :

Instances of `java.util.Random` are not cryptographically secure. Consider instead using `SecureRandom` to get a cryptographically secure pseudo-random number generator for use by security-sensitive applications.

Figure 15 : <https://docs.oracle.com/javase/7/docs/api/java/util/Random.html>

### II. Analyse dynamique :

#### 1. SSL Pinning Bypass :

L'application **NordPass** utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques *MITM (Man In The Middle)* dans des canaux de communication sécurisés. Ainsi, pour pouvoir intercepter et analyser le trafic, il faut contourner le *SSL Pinning*.

Pour ce faire, il faut, tout d'abord, télécharger et déployer le package du serveur Frida selon l'architecture de notre émulateur Android.

frida-server-16.0.7-android-arm.xz	6.62 MB
frida-server-16.0.7-android-arm64.xz	14.9 MB
frida-server-16.0.7-android-x86.xz	15 MB
frida-server-16.0.7-android-x86_64.xz	30.2 MB

Figure 16 : <https://github.com/frida/frida/releases>

Pour connaître l'architecture de l'appareil, exécutez la commande suivante :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell getprop ro.product.cpu.abi  
x86
```

Maintenant qu'on sait l'architecture utilisé (x86), nous pouvons télécharger le serveur '**frida-server-16.0.7-android-x86.xz**'

```
[(kali㉿kali)-[~]]  
└─$ wget https://github.com/frida/frida/releases/download/16.0.7/frida-server-16.0.7-android-x86.xz
```

Ensuite, nous allons le décompresser et le renommer en '**frida-server**'

```
[(kali㉿kali)-[~]]  
└─$ unxz frida-server-16.0.7-android-x86.xz  
  
[(kali㉿kali)-[~]]  
└─$ mv frida-server-16.0.7-android-x86 frida-server
```

Puis, nous allons transférer le package frida-server à notre émulateur Android en exécutant la commande suivante :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb push C:\Users\pc\Desktop\frida-server /data/local/tmp  
C:\Users\pc\Desktop\frida-server: 1 file pushed, 0 skipped. 1.5 MB/s (53604060 bytes in 34.370s)
```

Ensuite, nous allons attribuer les permissions d'exécution :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell chmod 777 /data/local/tmp/frida-server
```

Finalement, pour pouvoir exécuter Objection, il faut démarrer le serveur Frida en exécutant la commande :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell /data/local/tmp/frida-server &  
-
```

## App2 - NordPass

Avant d'utiliser Objection, il faut démarrer l'application NordPass puis exécuter la commande ci-dessous afin d'obtenir le nom du package :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>frida-ps -Uai
  PID  Name           Identifier
4 -----
2514  Email          com.android.email
2746  Keeper          com.callpod.android_apps.keeper
2576  Messaging       com.android.messaging
2778  NordPass        com.nordpass.android.app.password.manager
1954  Phone          com.android.dialer
1103  Settings        com.android.settings
2721  Superuser       com.genymotion.superuser
- Amaze            com.amaze.filemanager
- Calendar         com.android.calendar
- Camera            com.android.camera2
- Clock              com.android.deskclock
- Contacts          com.android.contacts
- Custom Locale    com.android.customlocale2
- Development Settings com.android.development_settings
- Files              com.android.documentsui
- Gallery            com.android.gallery3d
- Search              com.android.quicksearchbox
- WebView Shell     org.chromium.webview_shell
```

A ce niveau, nous pouvons exécuter Objection qui va nous permettre de contourner le *SSL Pinning* :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>objection --gadget com.nordpass.android.app.password.manager explore
Using USB device `Nexus 4`  
Agent injected and responds ok!
```



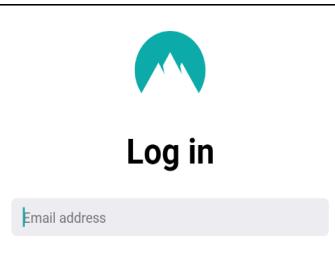
```
Runtime Mobile Exploration  
by: @leonjza from @sensepost
```

```
[tab] for command suggestions  
. . . android.app.password.manager on (Google: 10) [usb] # android sslpinning disable  
(agent) Custom TrustManager ready, overriding SSLContext.init()  
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.verifyChain()  
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.checkTrustedRecursive()  
(agent) Registering job 414302. Type: android-sslpinning-disable  
. . . android.app.password.manager on (Google: 10) [usb] #
```

➔ SSL Pinning est contourné avec succès

## 2. Interception du trafic :

### ▪ Lancement de l'application :



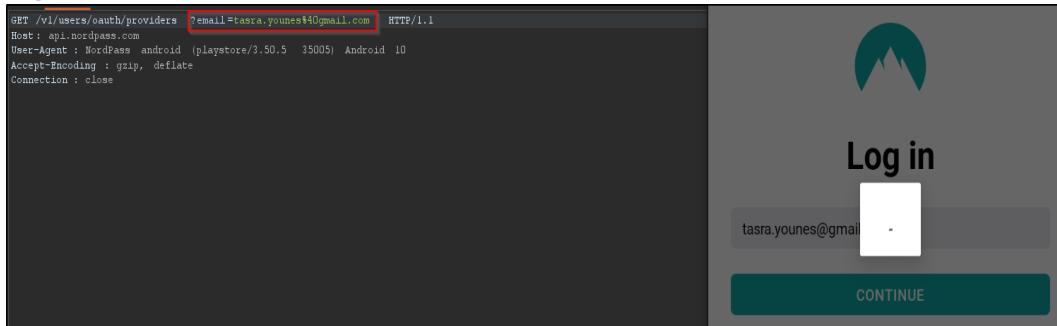
```
POST /y1/projects/9e441834858/namespaces.firebaseiofetch HTTP/2
Host: firebasehostconfig.googleapis.com
If-Match :etag-9e441834858-firebase-fetch-1822576912
X-Google-Auth-User :eyJhbGciOiIzUzI1InR5cC1I6IpXCVQJ-eyJhcHBfZC16IjE6OTX0NDE4MzQ4NTY4cmFuZHVvaWQ6YmJ1ZDYUZjAwNTAxNjRhMzKxMDU2ZC1sImV4cc16MTY4MTg0NTY1MywzImlkjIzIv9TqnhNgTdxL1TpkhNsXWlYIisInByzb2plI3RodwizX1icjkNDQxDMDQDUOHO.AEZLPV8wRgihAnNvf-OD_Ityazptbg-VriTyg-XBgqsgElUTnfczAEahmsZ2liiuReKaOrpmn82UStzRD9aGAJUZxsr_U
Content-Type : application/json
Accept : application/json
Content-Length : 688
User-Agent : Dalvik/2.1.0 (Linux; U; Android 10; Nexus 4 Build/QQID.200105.002)
Connection : Keep-Alive
Accept-Encoding : gzip, deflate
```

```
{
  "appInstanceId": "e_m8x4jSwyhQJDz0l0IXh",
  "appVersion": "2.50.5",
  "countryCode": "US",
  "analyticsUserProperties": {},
  "appId": "1:9e441834858:android:bbed64#0050164a791056d",
  "platformVersion": "2.9",
  "sdkVersion": "2.1.0.0",
  "packageName": "com.nordpass.android.app.password.manager",
  "appInstanceIdToken": "eyJhbGciOiIzUzI1InR5cC1I6IpXCVQJ-eyJhcHBfZC16IjE6OTX0NDE4MzQ4NTY4cmFuZHVvaWQ6YmJ1ZDYUZjAwNTAxNjRhMzKxMDU2ZC1sImV4cc16MTY4MTg0NTY1MywzImlkjIzIv9TqnhNgTdxL1TpkhNsXWlYIisInByzb2plI3RodwizX1icjkNDQxDMDQDUOHO.AEZLPV8wRgihAnNvf-OD_Ityazptbg-VriTyg-XBgqsgElUTnfczAEahmsZ2liiuReKaOrpmn82UStzRD9aGAJUZxsr_U",
  "languageCode": "en-US",
  "appBuildId": "33005"
}
```

## App2 - NordPass

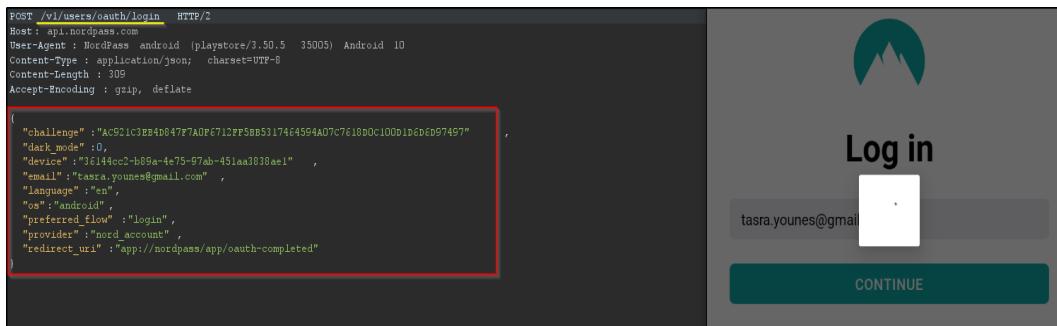
- ➔ Lors du démarrage de l'application, quelques métadonnées sont envoyées en clair comme : TimeZone, SDKversion, PackageName, LanguageCode, platformVersion, ...

### ▪ Signup :



A screenshot of the NordPass mobile application's login screen. On the left, there is a placeholder text field for an email address, with the placeholder text "tasra.younes@gmail.com". On the right, there is a "Log in" button.

- ➔ L'email est envoyé en clair sous forme d'un paramètre d'une requête HTTP GET



A screenshot of the NordPass mobile application's login screen. On the left, there is a placeholder text field for an email address, with the placeholder text "tasra.younes@gmail.com". On the right, there is a "Log in" button.

- ➔ Des métadonnées sont envoyées en clair sous format json comme : Email, os, language, redirect\_uri, ...



A screenshot of the NordPass mobile application's login screen. On the left, there is a placeholder text field for an email address, with the placeholder text "tasra.younes@gmail.com". On the right, there is a "Log in" button.

- ➔ Redirection vers :

<https://api.nordpass.com/v1/users/oauth/loginredirect?attempt=b604cc7c-7d1b-444e-9a1f-8be117d60022>

## App2 - NordPass

### La saisie du mot de passe :

The screenshot shows a NordPass login interface. On the left is a redacted terminal window displaying a POST request to '/login/password'. The password field in the request body is highlighted with a red box and contains the value 'password=letmein123'. To the right is the NordPass login page, which includes a 'Hi there!' greeting, a user input field with the email 'tasra.younes@gmail.com', a 'Log in' button, and a 'Forgot your password?' link.

```
POST /login/password?challenge=047cfe6d5573fcad4fd-b5b590e4fb63000 HTTP/2
Host: nordaccount.com
Cookie: sessions_bag =
MTI1MTDmNg==YmRhdicQmfjkRwShad0hf2ZdQKf9R0WV12X0nFFTtV8Q0FL211DQVWdipQX1E0k17TmbwPfHhmoWVZ1mlbMfB9f0mEaTA1WpgrHf5;
User-Agent: Firefox/94.0mpCUNy5clmKarf32PRAKlipcR9tcmJ7sq+qyXb6w=; nextId=00ec783-7d8c-4d0c-9402-f512acb6f170eb; dark_mode=false; locale=en; client_dimension=true; fontsCssCache=true; _ga=GAI.0.86948113.1671240887; _gaID=GAI.0.160351895.1671240887; _cf_lmt=1671240887
Accept: application/json, text/plain, */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Linux; Android 10; Nexus 8/01D; Build/QJ1.200105.002; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/74.0.3729.186 Mobile Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: https://nordaccount.com/login/password?challenge=047cfe6d5573fcad4fd-b5b590e4fb63000
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
X-Requested-With: org.chromium.webview_shell
password=letmein123 &_csrf=J3nkei1dFF5nGWAQ7HWvMdVNkt6p
```

### Mot de passe est envoyé en clair

Un attaquant peut simplement intercepter ce paquet et aisément lire le mot de passe de l'utilisateur lors de l'authentification à NordPass

#### ■ Crédit du Master Password :

The image displays three screenshots of NordPass API requests for creating a master password, each with a redacted terminal window showing the password in the request body.

- Screenshot 1:** A redacted terminal window shows a POST request to '/v2/users/identities'. The password 'Master\$Password1234\$' is visible in the request body.
- Screenshot 2:** A redacted terminal window shows a POST request to '/v2/users/actions'. The password 'Master\$Password1234\$' is visible in the request body.
- Screenshot 3:** A redacted terminal window shows a POST request to '/v2/users/actions'. The password 'Master\$Password1234\$' is visible in the request body.

Each screenshot also includes a 'Confirm Master Password' UI component with a 'Log Out' link and a 'Forgot your password?' link.

➔ Master Password créée avec succès.

## App2 - NordPass

### ■ Création d'un mot de passe :

```
POST /v2/users/identities/items HTTP/2
Host : api.nordpass.com
Authorization : Bearer acdall4fc8505007ed4d8df4ff919173cd4be862899fd537b69bd5+4d55
User-Agent : NordPass android (playstore/3.50.5 35005) Android 10
Content-Type : application/json; charset=UTF-8
Content-Length : 2054
Accept-Encoding : gzip, deflate

[{"change": {"x2PyIaDyfM0t7WqS9hEHPXnaiBViobcNc20nLrs/iD/oTuRGK1E=SSd247qfQUCPq09kx69V677Hy01T1IaItDUWHmyg/u003d/u003d", "change_editor_signature": "MSAOzX0pp+f7AC5t8b5ckAv7og854+08002+fiChFLXNDfC4pym5M6C3kj5+qheXa1QSXrH8FmfifzUaw/u003d/u003d", "dek_info": "X5alsm0Poly105-X5510", "editor_signature": "CxRcMcTxzEIM0+07UDxSzLqfNcSC/P5rrl24+vi1hpHQjPTC718+g6/IMY31NqfnKK7ayPFTYObRSaBQ/u003d/u003d", "encrypted_value_key": "f5fc5415-acda-4b07-84e4-37225a400a10", "item_id": "X5alsm0Poly105-X5510", "item_key": {"dek_info": "X5alsm0Poly105-X5510", "encrypted_item_private_key": "xTCfCMcmWtpfGLRktq1W0G2BsfyPfTSK1ldktihOyphb7C1Qm5hVscHe0TfTfRnPMwUfuQXW61Rf6qd1llFoxt9Rxp5+p4A/YcNigVFRhTjUGVphCmBhse84f/C0d12M9gAcneCcfvH4dJdLMMHLS/5AhTfJqAfTf71KT7y2Hg/u003d/u003d", "item_id": "f70111ed-c59d-44ca-bd39-6f5f83474b30c", "item_signature": "f14+ad3Ac8501kstedfswMy0URGSTw+99e-Erq8Tf3x7ZPp8pVnBEDhF4UishXK57Pw7gkTYB02UanCg/u003d/u003d", "shared_signature": "7ym/449yFAul+AdoCf/OIFQlfVt515kgQunIVK0WcXBTofftbXk6zKA3+s79bRa7GEBS15zc/WJTpfrcqZag/u003d/u003d", "version": "1.0.0"}, {"public_key": "+1jnl1hdB0gNm/EDkJDa3Sj5G5w1l4RxBXqudtB/u003d", "type": "item", "value": "AsuKrf3J4xChNeh41lyvNSMfh5TbW3mHPI14+Rfxwcfal+kjg0AzIAfeqolk1h0fe8Y3In/16WueOchsykElmyOlaZD0+c5D14X2+hTospHgQ01dcVx4rx7uJ0h5z2VygfNhr+8mpy6c9p4x13iUTTfmg13XfvfVNQdmfbd1AbAx+rx15kRV/1fNveEMw1f1pmzjjUctUgsfJQVNS/CLXjsocYD6szjX37bxz6uSAU2Oz6e201bhyg/yjfd7Utrfem8x4uf0fyJSVjV2D057WmK130atdeBttxo/qAA8k8z+QHADAXfLcpg1Qjb/bsga1L+gvPh/n/V3118f6Gmdovs1kvmqIdr7Mfnb#9502a4eeFVxjdfffP3v2gSMEx0Xvrdw0in0050jpaAI/c0LE+jSS7hn9Bw2+Ubhg7q7cq-g/u003d/u003d", "value_signature": "7ym/Fx1v1h2u67F9Bpmc1KfRe51+7yZMSE74c671tQhcb394e1QdhgtSH40tEsF0nXfW6skycWmBa51drvh3pg/u003d/u003d", "value_version": "2021-12-17 19:45:55"}]
```

```
POST /v1/users/onboarding/progress HTTP/2
Host : api.nordpass.com
Authorization : Bearer acdall4fc8505007ed4d8df4ff919173cd4be862899fd537b69bd5+4d55
User-Agent : NordPass android (playstore/3.50.5 35005) Android 10
Content-Type : application/json; charset=UTF-8
Content-Length : 28
Accept-Encoding : gzip, deflate

{"tasks": [{"password_added": true}], "status": "success"}
```

The screenshot shows the NordPass mobile application interface. At the top, there is a title bar with the Instagram logo and the text 'Title\* Instagram.com'. Below this is a section labeled 'Password Details' with fields for 'Email or Username' (set to 'hacker') and 'Password' (set to '5nj^0E1CnM0hgnIkYTgb'). A large green button labeled 'GENERATE PASSWORD' is visible. In the middle of the screen, there is a red box highlighting the 'password\_added' field in the JSON response from the API call. The JSON response includes fields like 'title', 'password', 'status', and 'tasks'. The 'tasks' array contains an object with 'password\_added' set to true. At the bottom, there is another section labeled 'Other'.

➔ Un mot de passe robuste est généré par NordPass et créé avec succès.

## IV. Conclusion :

Properties		
<b>Dangerous permissions</b>		3
<b>Unknown permissions</b>		3
<b>Encryption Standard</b>		XChaCha20
<b>Secure Hashing Algorithms</b>		(✗)
<b>Secure PRG</b>		(✗)
<b>Zero Knowledge Architecture</b>		(✓)
<b>2FA Authentication</b>		(✓)
<b>MFA Authentication</b>		(✓)
<b>Biometrics</b>		(✓)
<b>Automatic form-filling</b>		(✓)
<b>Security alerts</b>		(✗)
<b>Mobile application</b>		(✓)
<b>Security audits</b>		(✗)
<b>Import of passwords</b>		(✓)
<b>Export of passwords</b>		(✓)
<b>SSO Authentication</b>		(✗)
<b>Password sharing</b>		(✓)
<b>Integrated database</b>		(✗)
<b>Certificate Pinning</b>		(✓)
<b>Root detection</b>		(✓)
<b>Encrypted traffic</b>		(✗)
<b>Encrypted email</b>		(✗)
<b>Encrypted Master Password</b>		(✗)

## App 3 - LastPass:



Figure 17: LastPass <https://www.lastpass.com/>

**LastPass** est l'un des gestionnaires de mots de passe les plus simples et les plus riches en fonctionnalités. Il dispose d'une interface simple et sécurisée, d'un chiffrement fort et de multitudes de fonctionnalités comme le partage de mots de passe, l'authentification à deux facteurs (*Two-Factor Authentication*) et l'héritage numérique (Digital legacy). Cependant, des violations de données et d'autres problèmes de sécurité ont été signalés pour ce service. De plus, LastPass n'offre pas le même niveau de contrôle sur vos données que d'autres gestionnaires de mots de passe.

### I. Analyse statique :

#### 1. Permissions de l'application :

##### ▪ Permissions dangereuses :

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

Cette permission permet à l'application d'utiliser les fonctionnalités d'authentification de compte du gestionnaire de compte, y compris la création de nouveaux comptes ainsi que l'obtention et la configuration de leurs mots de passe.

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

Cette permission permet à l'application de prendre des photos des vidéos et de collecter les images que la caméra peut voir à tout moment.

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

Cette permission permet à l'application d'accéder à la liste des comptes du *Accounts Service*.

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

Cette permission permet à l'application d'effectuer des opérations comme l'ajout et la suppression de comptes et la suppression de leur mot de passe.

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

Cette permission permet à l'application de lire à partir d'un stockage externe.

## App3 - LastPass

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

Cette permission permet à l'application de demander des jetons d'authentification (*Authentication Tokens*)

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

Cette permission permet à l'application de lire/modifier/supprimer le contenu du stockage externe.

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

Cette permission permet à l'application d'installer de nouveaux package Android et mises à jour. Des applications malveillantes peuvent exploiter cette permission pour ajouter de nouvelles applications avec des autorisations élevées (*System-level permissions*)

- **Permissions inconnues :**

`<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>`  
`<uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>`

- **Permissions normales :**

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

Ces permissions permettent à l'application de :

- ➔ Afficher des informations sur l'état du Wi-Fi.
- ➔ Visualiser l'état du réseau.
- ➔ Créer des sockets réseau.

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

Ces permissions permettent à l'application de lire/modifier les paramètres de synchronisation

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

Cette permission permet à l'application d'empêcher le smartphone de se mettre en veille.

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

Cette permission permet à l'application de se lancer dès que le système a fini le démarrage. Cela peut rendre le démarrage du téléphone plus long et permettre à l'application de ralentir l'ensemble du téléphone en étant toujours en cours d'exécution.

## 2. Protection des données :

**LastPass** est un fournisseur de sécurité 'Zero-Knowledge'. En effet, le chiffrement se déroule sur les ordinateurs ou les smartphones avant d'être envoyé sur les serveurs du logiciel.

## App3 - LastPass

Autrement dit, personne, y compris l'équipe **LastPass**, ne peut voir les mots de passe stockés dans l'application.

LastPass utilise l'algorithme de chiffrement **AES-256 bits** pour chiffrer et protéger le contenu du coffre-fort numérique LastPass.

LastPass utilise PBKDF2-SHA256 avec plus de 100.000 tours

```
public class SecureCoderDecoder {  
    private static final String AES_KEY_ALG = "AES"; ←  
    private static final String BACKUP_PBE_KEY_ALG = "PBEWithMD5AndDES";  
    private static final int ITERATIONS = 2000;  
    private static final int KEY_SIZE = 256; ←  
    private static final String PRIMARY_PBE_KEY_ALG = "PBKDF2WithHmacSHA1"; ←  
    private static final String PROVIDER = "BC";  
    private static final String TAG = "SecureCoderDecoder";  
    private byte[] sKey;
```

### 3. Algorithmes cryptographiques non sécurisés :

L'application **LastPass** utilise un algorithme cryptographique non sécurisé à savoir MD5, qui entraîne des collisions de hachage.

Ces collisions signifient qu'un attaquant pourrait créer des fichiers qui auraient exactement le même hachage qu'un autre, ce qui rend impossible que le fichier n'a pas été falsifié.

```
public static String computeMd5(File file) {  
    long currentTimeMillis = System.currentTimeMillis();  
    byte[] bArr = new byte[CacheHelper.VALUE_TO_CONVERT_MB_TO_BYTES];  
    try {  
        MessageDigest messageDigest = MessageDigest.getInstance("MD5");  
        FileInputStream fileInputStream = new FileInputStream(file);  
        while (true) {  
            int read = fileInputStream.read(bArr);  
            if (read <= 0) {  
                break;  
            }  
            messageDigest.update(bArr, 0, read);  
        }  
        byte[] digest = messageDigest.digest();  
        String bigInteger = new BigInteger(1, digest).toString(16);  
        fileInputStream.close();  
        if (bigInteger.length() != 33) {  
            String str = "";  
            for (int i = 1; i < 33 - bigInteger.length(); i++) {  
                str = str.concat("0");  
            }  
            bigInteger = str.concat(bigInteger);  
        }  
        Logger logger = Logger.getInstance();  
        String str2 = TAG;  
        logger.m26062v(str2, "computeMd5: duration: " + (System.currentTimeMillis() - currentTimeMillis) + " ms");  
        return bigInteger;  
    } catch (Exception e) {  
        e.printStackTrace();  
        return null;  
    }  
}  
  
private static String m2731j(String str) {  
    try {  
        byte[] digest = MessageDigest.getInstance("MD5").digest(str.getBytes("UTF-8"));  
        StringBuilder sb = new StringBuilder(digest.length * 2);  
        for (byte b : digest) {  
            int i = b & 255;  
            if (i < 16) {  
                sb.append("0");  
            }  
            sb.append(Integer.toHexString(i));  
        }  
        return sb.toString();  
    } catch (UnsupportedEncodingException e) {  
        throw new RuntimeException("Huh, UTF-8 should be supported?", e);  
    } catch (NoSuchAlgorithmException e2) {  
        throw new RuntimeException("Huh, MD5 should be supported?", e2);  
    }  
}
```

LastPass utilise aussi l'algorithme de hachage SHA-1 qui entraîne aussi des collisions de hachage.

## App3 - LastPass

```
private static void installLinuxPRNGSecureRandom() throws SecurityException {
    if (Build.VERSION.SDK_INT > 18) {
        return;
    }
    Provider[] providers = Security.getProviders("SecureRandom.SHA1PRNG");
    if (providers == null || providers.length < 1 || !LinuxPRNGSecureRandomProvider.class.equals(providers[0].getClass())) {
        Security.insertProviderAt(new LinuxPRNGSecureRandomProvider(), 1);
    }
    SecureRandom secureRandom = new SecureRandom();
    if (LinuxPRNGSecureRandomProvider.class.equals(secureRandom.getProvider().getClass())) {
        try {
            SecureRandom secureRandom2 = SecureRandom.getInstance("SHA1PRNG");
            if (LinuxPRNGSecureRandomProvider.class.equals(secureRandom2.getProvider().getClass())) {
                return;
            }
            throw new SecurityException("SecureRandom.getInstance(\"SHA1PRNG\") backed by wrong Provider: " + secureRandom2.getProvider().getClass());
        } catch (NoSuchAlgorithmException e) {
            throw new SecurityException("SHA1PRNG not available", e);
        }
    }
    throw new SecurityException("new SecureRandom() backed by wrong Provider: " + secureRandom.getProvider().getClass());
}

public static String computeHmacSha1(String str, String str2) {
    try {
        SecretKeySpec secretKeySpec = new SecretKeySpec(str2.getBytes("UTF-8"), "HmacSHA1");
        Mac mac = Mac.getInstance("HmacSHA1");
        mac.init(secretKeySpec);
        return convToHex(mac.doFinal(str.getBytes("UTF-8")));
    } catch (UnsupportedEncodingException e) {
        Logger.getInstance().e(TAG, (Throwable) e);
        return "";
    } catch (InvalidKeyException e2) {
        Logger.getInstance().e(TAG, (Throwable) e2);
        return "";
    } catch (NoSuchAlgorithmException e3) {
        Logger.getInstance().e(TAG, (Throwable) e3);
        return "";
    }
}
```

### 4. Stockage externe :

L'application **LastPass** a les permissions de lecture/écriture sur le stockage externe, ce qui signifie que toute application malveillante installée sur le smartphone peut lire les données enregistrées sur le stockage externe.

```
protected static boolean m1686i() {
    return Environment.getExternalStorageState().equals("mounted") && !Environment.isExternalStorageEmulated();
}

public class DefaultInstaller implements Installer {
    public static final String OBB_FOLDER = Environment.getExternalStorageDirectory().getAbsolutePath() + "/Android/obb/";
    private static final String TAG = DefaultInstaller.class.getSimpleName();
    private final AppInstaller appInstaller;
    private final AppInstallerStatusReceiver appInstallerStatusReceiver;
    private final AptoideInstalledAppsRepository aptoideInstalledAppsRepository;
```

### 5. Générateurs de nombres aléatoires non sécurisés :

L'application **LastPass** utilise un générateur de nombres aléatoires non sécurisé `java.util.Random`

## App3 - LastPass

```
import java.util.Random;

/* compiled from: RandomEventSampler.java */
/* Loaded from: classes2.dex */
public class l implements e {
    private double a;
    private Random b;

    public l(double d) {
        this(d, new Random());
    }

    @Override // io.sentry.connection.e
    public boolean a(q.b.l.b bVar) {
        return this.a >= Math.abs(this.b.nextDouble());
    }

    public l(double d, Random random) {
        this.a = d;
        this.b = random;
    }
}
```

### 6. Root detection :

L'objectif de *Root Detection* est de rendre l'exécution d'une application sur un appareil rooté un peu plus difficile, ce qui empêche l'utilisation des outils et techniques de *reverse engineering*.

L'application **LastPass** implémente le *Root Detection*

```
protected static Boolean j() {
    String str = Build.TAGS;
    if (str != null && str.contains("test-keys")) {
        return true;
    }
    String[] strArr = {"/data/local/bin/su", "/data/local/su", "/data/local/xbin/su", "/sbin/su", "/su/bin", "/su/bin/su"};
    for (int i = 0; i < 15; i++) {
        try {
        } catch (RuntimeException e2) {
            Log.e(b, "Exception while attempting to detect whether the device is rooted", e2);
        }
        if (new File(strArr[i]).exists()) {
            return true;
        }
    }
    return false;
}
```

### 7. SQLite database :

L'application utilise une base de données SQLite et exécute des requête SQL non sécurisées. Une saisie non vérifiée dans ses requêtes SQL peut provoquer une *SQL injection*. Les informations sensibles doivent également être cryptées dans la base de données.

```
import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.util.SparseArray;

/* compiled from: DefaultDatabaseImpl.java */
/* Loaded from: classes2.dex */
class h implements g {
    private final SparseArray<com.liulishuo.filedownloader.model.a> b = new SparseArray<>();
    private final SQLiteDatabase a = new c(n.h.a.f0.c.a()).getWritableDatabase();

    public h() {
        a();
    }
}
```

## App3 - LastPass

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/* compiled from: DefaultDatabaseOpenHelper.java */
/* loaded from: classes2.dex */
class c extends SQLiteOpenHelper {
    public c(Context context) {
        super(context, "filedownloader.db", (SQLiteDatabase.CursorFactory) null, 2);
    }

    @Override // android.database.sqlite.SQLiteOpenHelper
    public void onCreate(SQLiteDatabase sQLiteDatabase) {
        sQLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS filedownloader( _id INTEGER PRIMARY KEY, url VARCHAR, path VARCHAR, status TINYINT(7), sofar INTEGER, ");
    }

    @Override // android.database.sqlite.SQLiteOpenHelper
    public void onUpgrade(SQLiteDatabase sQLiteDatabase, int i, int i2) {
        if (i == 1 && i2 == 2) {
            sQLiteDatabase.execSQL("ALTER TABLE filedownloader ADD COLUMN pathAsDirectory TINYINT(1) DEFAULT 0");
            sQLiteDatabase.execSQL("ALTER TABLE filedownloader ADD COLUMN filename VARCHAR");
        }
    }
}
```

## **8. Certificate Pinning :**

L'application **LastPass** utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques *MITM (Man In The Middle)* dans des canaux de communication sécurisés.

```
private static X509TrustManager a() {
    TrustManagerFactory trustManagerFactory;
    try {
        trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        try {
            trustManagerFactory.init((KeyStore) null);
        } catch (KeyStoreException e) {
            e = e;
            d1.a("SslPinningValidator", "Error in getting trust manager: ", e);
            if (trustManagerFactory == null) {
            }
        } catch (NoSuchAlgorithmException e2) {
            e = e2;
            d1.a("SslPinningValidator", "Error in getting trust manager: ", e);
            if (trustManagerFactory == null) {
            }
        }
    } catch (KeyStoreException e3) {
        e = e3;
        trustManagerFactory = null;
    } catch (NoSuchAlgorithmException e4) {
        e = e4;
        trustManagerFactory = null;
    }
    if (trustManagerFactory == null) {
        return null;
    }
    TrustManager[] trustManagers = trustManagerFactory.getTrustManagers();
    for (TrustManager trustManager : trustManagers) {
        if (trustManager instanceof X509TrustManager) {
            return (X509TrustManager) trustManager;
        }
    }
    return null;
}
```

## App3 - LastPass

### 9. Divulgation des adresses IP :

L'application LastPass divulgue des adresses IPv4 et IPv6 :

```
private void a(k kVar) throws IOException {
    if (this.f == null) {
        if (kVar.e() instanceof Inet6Address) {
            this.f = InetAddress.getByName("FF02::FB");
        } else {
            this.f = InetAddress.getByName("224.0.0.251");
        }
    }
    if (this.g != null) {
        U();
    }
    this.g = new MulticastSocket(r.a.g.s.a.a);
    if (kVar != null && kVar.f() != null) {
        try {
            this.g.setNetworkInterface(kVar.f());
        } catch (SocketException e2) {
            if (z.a()) {
                t.b.b bVar = z;
                bVar.b("openMulticastSocket() Set network interface exception: " + e2.getMessage());
            }
        }
    }
    this.g.setTimeToLive(255);
    this.g.joinGroup(this.f);
```

### 10. SafetyNet API :

SafetyNet est une API Android qui fournit un ensemble de services et crée des profils de dispositifs en fonction d'information sur le logiciel et le matériel. Ce profil est ensuite comparé à une liste de modèles de dispositifs approuvés qui ont passé les tests de compatibilité Android. SafetyNet tente également de détecter si l'appareil est rooté.

```
import com.google.android.gms.safetynet.SafetyNetApi;
import com.google.android.gms.safetynet.SafetyNetClient;

public class zzk implements SafetyNetApi {
    /* JADY INFO: Access modifiers changed from: package-private */
    /* Loaded from: classes2.dex */
    public static abstract class a extends com.google.android.gms.internal.safetynet.a<SafetyNetApi.zzb> {
        /* renamed from: s reason: collision with root package name */
        protected final zzg f2659s;

        public a(GoogleApiClient googleApiClient) {
            super(googleApiClient);
            this.f2659s = new c(this);
        }

        /* JADY INFO: Access modifiers changed from: protected */
        @Override // com.google.android.gms.common.api.internal.BasePendingResult
        public /* synthetic */ Result a(Status status) {
            return new b(status, null);
        }
    }
}
```

## II. Analyse dynamique :

### 1. SSL Pinning Bypass :

L'application **LastPass** utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques *MITM* (*Man In The Middle*) dans des canaux de communication sécurisés. Ainsi, pour pouvoir intercepter et analyser le trafic, il faut contourner le *SSL Pinning*.

Pour ce faire, nous allons suivre les mêmes étapes qu'on a fait pour contourner le *SSL Pinning* implémenté dans l'application NordPass.

## App3 - LastPass

- 1) Télécharger le serveur Frida (*x86 Architecture*) :

```
[~] kali㉿kali ~ $ wget https://github.com/frida/frida/releases/download/16.0.7/frida-server-16.0.7-android-x86.xz
```

- 2) Décompresser et renommer en ‘frida-server’ :

```
[~] kali㉿kali ~ $ unxz frida-server-16.0.7-android-x86.xz  
[~] kali㉿kali ~ $ mv frida-server-16.0.7-android-x86 frida-server
```

- 3) Transférer le package ‘frida-server’ à l’émulateur Android :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb push C:\Users\pc\Desktop\frida-server /data/local/tmp  
C:\Users\pc\Desktop\frida-server: 1 file pushed, 0 skipped. 1.5 MB/s (53604060 bytes in 34.370s)
```

- 4) Attribuer les permissions d’exécution au package :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell chmod 777 /data/local/tmp/frida-server
```

- 5) Démarrer le serveur Frida :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell /data/local/tmp/frida-server &  
-
```

- 6) Obtenir le nom du package de l’application :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>frida-ps -Uai  
 PID Name Identifier  
 4 -----  
 5085 Aptoide cm.aptoide.pt  
 4816 Email com.android.email  
 5214 Keeper com.callpod.android_apps.keeper  
 5481 LastPass com.lastpass.lpandroid ←  
 4895 Messaging com.android.messaging  
 5245 NordPass com.nordpass.android.app.password.manager  
 4587 Phone com.android.dialer  
 5015 Settings com.android.settings  
 5059 Superuser com.genymotion.superuser  
 - Amaze com.amaze.filemanager  
 - Calendar com.android.calendar  
 - Camera com.android.camera2  
 - Clock com.android.deskclock  
 - Contacts com.android.contacts  
 - Custom Locale com.android.customlocale2  
 - Development Settings com.android.development_settings  
 - Files com.android.documentsui  
 - Gallery com.android.gallery3d  
 - Search com.android.quicksearchbox  
 - WebView Shell org.chromium.webview_shell
```

- 7) Exécuter Objection pour contourner le *SSL Pinning* :

## App3 - LastPass

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>objection --gadget com.lastpass.lpandroid explore  
Using USB device `Nexus 4`  
Agent injected and responds ok!  
  
[object]inject(i)on v1.11.0  
  
Runtime Mobile Exploration  
by: @leonjza from @sensepost  
  
[tab] for command suggestions  
com.lastpass.lpandroid on (Google: 10) [usb] # android ssplining disable  
(agent) Custom TrustManager ready, overriding SSLContext.init()  
(agent) Found okhttp3.CertificatePinner, overriding CertificatePinner.check()  
(agent) Found okhttp3.CertificatePinner, overriding CertificatePinner.check$okhttp()  
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.verifyChain()  
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.checkTrustedRecursive()  
(agent) Registering job 338664. Type: android-ssplining-disable  
com.lastpass.lpandroid on (Google: 10) [usb] #
```

→ SSL Pinning est contourné avec succès

### 3. Interception du trafic :

#### ▪ Signup

La saisie de l'adresse email :

```
POST /create_account.php ?check=avail&username=tasra.younes@gmail.com &lang=en_US &mimetype=1 HTTP/2  
Host : lastpass.com  
Content-Type : application/x-www-form-urlencoded  
Content-Length : 88  
Accept-Encoding : gzip, deflate  
User-Agent : okhttp/3.10.0  
  
requestsrc = android &method = android &hasplugin = 4.11.7 &lpversion = 4.11.7 &sentms = 1671333006716
```

Let's get started!

Email address

tasra.younes@gmail.com

→ L'adresse email est envoyé en clair sous forme d'un paramètre d'une requête HTTP GET

La création du Master Password :

```
POST /create_account.php HTTP/2  
Host : lastpass.com  
Content-Type : application/x-www-form-urlencoded  
Content-Length : 378  
Accept-Encoding : gzip, deflate  
User-Agent : okhttp/3.10.0  
  
requestsrc = android &method = android &hasplugin = 4.11.7 &lpversion = 4.11.7 &sentms = 1671333100198 &timezone2 = -0530AD0E0C1 &logins = 14  
language2 = en_US &xm = 1&contactPermission = 1&marketingOptOutSeen = 0&email = tasra.younes@gmail.com &iterations = 100000 &hash =  
e8b110f44f68b77fcfcfc288d1fcdb1520ff1ebfcfc0255b77bfcd198c4 &password_hint = Master &deviceType = phone &username =  
tasra.younes@gmail.com
```

Now we'll create your master password

Master Password

MasterPassword\$

Please wait

Confirm master password

MasterPassword\$

Password hint (optional)

Master

→ Le Master Password est envoyé en clair sous forme de Hash

Par contre le 'Password Hint' est envoyé en clair.

## App3 - LastPass

#### ▪ Se connecter à LastPass :



Your vault is ready!

Items you save in LastPass are stored safely in your vault.

Enjoy priority support and more with a 1-day free trial of Premium, on us.

[GO TO MY VAULT](#)

L'adresse email est envoyé en clair et le Master Password est envoyé sous forme de Hash. Autrement dit, si le mot de passe n'est pas fort, un attaquant pourrait facilement cracker le Hash en utilisant des outils comme *Hashcat* ou *JohnTheRipper* et récupérer le mot de passe de l'utilisateur et il aura éventuellement accès aux données stockées dans le coffre-fort numérique de LastPass.

Le Hash est de type ***SHA-256*** :

### III. Conclusion :

Properties	LastPas ...
Dangerous permissions	8
Unknown permissions	2
Encryption Standard	AES-256
Key Expansion Algorithm	PBKDF2-SHA256
Secure Hashing Algorithms	(✗)
Secure PRG	(✗)
Zero Knowledge Architecture	(✓)
2FA Authentication	(✓)
MFA Authentication	(✓)
Biometrics	(✓)
Automatic form-filling	(✓)
Security alerts	(✓)
Mobile application	(✓)
Security audits	(✓)
Import of passwords	(✓)
Export of passwords	(✓)
SSO Authentication	(✓)
Password sharing	(✓)
Integrated database	(✓)
Certificate Pinning	(✓)
Root detection	(✓)
Encrypted traffic	(✗)
Encrypted email	(✗)
Encrypted Master Password	(✓)

## App 4 - Bitwarden:



Figure 18: bitwarden <https://bitwarden.com/>

**Bitwarden** est un service moderne, libre et open source qui propose une application sur la plupart des plateformes et des navigateurs et se synchronise facilement partout.

**Bitwarden** propose un chiffrement des données de bout en bout, ce qui signifie que le client sera seul à connaître le contenu des données stockées. Il est même possible d'héberger soi-même le coffre-fort Bitwarden sur son propre serveur, pour pouvoir garantir soi-même la sécurité des données.

### I. Analyse statique :

#### 1. Permissions de l'application :

##### ▪ Permissions dangereuses :

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

Cette permission permet à l'application de prendre des photos des vidéos et de collecter les images que la caméra peut voir à tout moment.

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

Cette permission permet à l'application d'afficher des fenêtres d'alerte système. Les applications malveillantes peuvent contrôler l'écran du téléphone.

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

Cette permission permet à l'application de lire/modifier/supprimer le contenu du stockage externe.

##### ▪ Permissions inconnues :

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

```
<uses-permission android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
```

```
<uses-permission android:name="com.x8bit.bitwarden.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/>
```

## App4 - Bitwarden

### ▪ Permissions normales :

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

Ces permissions permettent à l'application de :

- ➔ Visualiser l'état du réseau.
- ➔ Créer des sockets réseau.

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

Cette permission permet à l'application de communiquer avec des étiquettes, cartes et des lecteurs NFC (*Near-Field Communication*)

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

Ces permissions permettent à l'application d'utiliser les modalités biométriques supportés par l'appareil.

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

Cette permission permet à l'application de contrôler le vibrateur.

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

Cette permission permet à l'application d'empêcher le téléphone de se mettre en veille.

## 2. Protection des données :

**Bitwarden** est un fournisseur de sécurité ‘Zero-Knowledge’. En effet, le chiffrement se déroule sur les ordinateurs ou les smartphones avant d'être envoyé sur les serveurs du logiciel.

Les mots de passes ainsi que les données privées sont verrouillés à l'aide d'un chiffrement de bout en bout, en utilisant **AES-256-CBC** avec un **salted hashing**, une fonction de dérivée **PBKDF2 HmacSHA-256**, ainsi qu'une fonction de padding **PKCS7**

```
public String getAlgorithm() {
    return "AES/CBC/PKCS7Padding/256/HmacSHA256";
}

@Override // com.microsoft.appcenter.utils.crypto.CryptoHandler
public void generateKey(ICryptoFactory cryptoFactory, String alias, Context context) throws Exception {
    Calendar calendar = Calendar.getInstance();
    calendar.add(1, 1);
    CryptoUtils.IKeyGenerator keyGenerator = cryptoFactory.getKeyGenerator("HmacSHA256", "AndroidKeyStore");
    keyGenerator.init(new KeyGenParameterSpec.Builder(alias, 4).setKeyValidityForOriginationEnd(calendar.getTime()).build());
    keyGenerator.generateKey();
}

public byte[] encrypt(ICryptoFactory cryptoFactory, int apiLevel, KeyStore.Entry keyStoreEntry, byte[] input) throws Exception {
    SecretKey secretKey = ((KeyStore.SecretKeyEntry) keyStoreEntry).getSecretKey();
    byte[] subkey = getSubkey(secretKey, 32);
    byte[] subkey2 = getSubkey(secretKey, 16);
    CryptoUtils.ICipher cipher = cryptoFactory.getCipher("AES/CBC/PKCS7Padding", null);
    cipher.init(1, new SecretKeySpec(subkey, "AES"));
    byte[] iv = cipher.getIV();
    byte[] doFinal = cipher.doFinal(input);
    byte[] macBytes = getMacBytes(subkey2, iv, doFinal);
    ByteBuffer allocate = ByteBuffer.allocate(iv.length + 1 + 1 + macBytes.length + doFinal.length);
    allocate.put((byte) iv.length);
    allocate.put(iv);
    allocate.put((byte) macBytes.length);
    allocate.put(macBytes);
    allocate.put(doFinal);
    return allocate.array();
}
```

## App4 - Bitwarden

### 3. Générateurs de nombres aléatoires non sécurisés :

L'application **Bitwarden** utilise un générateur de nombres aléatoires non sécurisé `java.util.Random`

```
import java.util.Random;
import java.util.concurrent.TimeUnit;

/* Loaded from: classes.dex */
public class HttpClientRetryer extends HttpClientDecorator {
    static final long[] RETRY_INTERVALS = {TimeUnit.SECONDS.toMillis(10), TimeUnit.MINUTES.toMillis(5), TimeUnit.MINUTES.toMillis(20)};
    private final Handler mHandler;
    private final Random mRandom;

    /* JADX INFO: Access modifiers changed from: package-private */
    public HttpClientRetryer(HttpClient decoratedApi) {
        this(decoratedApi, new Handler(Looper.getMainLooper()));
    }

    HttpClientRetryer(HttpClient decoratedApi, Handler handler) {
        super(decoratedApi);
        this.mRandom = new Random();
        this.mHandler = handler;
    }
}
```

### 4. SQLite database :

L'application **Bitwarden** utilise une base de données SQLite et exécute des requête SQL brutes (non vérifiées). Une saisie non vérifiée dans ses requêtes SQL peut provoquer une *SQL injection*. Une attaque par injection SQL réussie permet à l'attaquant de lire ou d'écrire dans la base de données et éventuellement exécuter des commandes administratives, en fonction des permissions accordées par le serveur.

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteFullException;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;

public DatabaseManager(Context context, String database, String defaultTable, int version, ContentValues schema, final String sqlCreateCommand, Listener listener) {
    this.mContext = context;
    this.mDatabase = database;
    this.mDefaultTable = defaultTable;
    this.mSchema = schema;
    this.mListener = listener;
    this.mSQLiteOpenHelper = new SQLiteOpenHelper(context, database, null, version) { // from class: com.microsoft.appcenter.utils.storage.DatabaseManager.1
        @Override // android.database.sqlite.SQLiteOpenHelper
        public void onCreate(SQLiteDatabase db) {
            db.execSQL(sqlCreateCommand);
            DatabaseManager.this.mListener.onCreate(db);
        }
    }
}
```

### 5. Utilisation du presse-papiers (Clipboard) :

Lorsque des données sont saisies, le presse-papiers peut être utilisé pour copier des données. Le presse-papiers est accessible à l'ensemble du système et est donc partagé par les applications. Ce partage peut être utilisé par des applications malveillantes pour obtenir des données sensibles qui enregistrées dans le presse-papiers.

L'application **Bitwarden** copie des données dans le presse-papiers, ce qui signifie qu'une application malveillante peut avoir accès à ces données.

```
import android.content.ClipboardManager;
import java.util.ArrayList;
import mono.android.IGCUserPeer;
import mono.android.Runtime;
import mono.android.TypeManager;

/* Loaded from: classes2.dex */
public class ClipboardManager_OnPrimaryClipChangedListenerImplementor implements IGCUserPeer, ClipboardManager.OnPrimaryClipChangedListener {
    public static final String __md_methods = "n_onPrimaryClipChanged():V:GetOnPrimaryClipChangedHandler:Android.Content.ClipboardManager/IOnPrimaryClipChangedListenerInvoker";
    private ArrayList refList;

    private native void n_onPrimaryClipChanged();

    static {
        Runtime.register("Android.Content.ClipboardManager+IOnPrimaryClipChangedListenerImplementor", Mono.Android, ClipboardManager_OnPrimaryClipChangedListenerImplementor.class);
    }

    public ClipboardManager_OnPrimaryClipChangedListenerImplementor() {
        if (getClass() == ClipboardManager_OnPrimaryClipChangedListenerImplementor.class) {
            TypeManager.Activate("Android.Content.ClipboardManager+IOnPrimaryClipChangedListenerImplementor, Mono.Android", "", this, new Object[0]);
        }
    }
}
```

## App4 - Bitwarden

### 6. Hardcoded sensitive information :

L'application stocke des données sensibles comme : APIKeys, Keys, usernames, passwords etc ...

```
public class DatabaseManager implements Closeable {
    public static final String PRIMARY_KEY = "oid";
    public static final String[] SELECT_PRIMARY_KEY = {PRIMARY_KEY};
    private final Context mContext;
}

public class Constants {
    public static boolean APPLICATION_DEBUGGABLE = false;
    public static final String APP_SECRET = "App-Secret";
    public static final String AUTHORIZATION_HEADER = "Authorization";
}

public class AppCenter {
    static final String APP_SECRET_KEY = "appsecret";
    static final String CORE_GROUP = "group_core";
}

public class OneCollectorIngestion extends AbstractAppCenterIngestion {
    static final String API_KEY = "apikey";
    private static final String CLIENT_VERSION_FORMAT = "ACS-Android-Java-no-%s-no";
    static final String CLIENT_VERSION_KEY = "Client-Version";
    private static final String CONTENT_TYPE_VALUE = "application/x-json-stream; charset=utf-8";
    private static final String DEFAULT_LOG_URL = "https://mobile.events.data.microsoft.com/OneCollector/1.0";
    static final String STRICT = "Strict";
    static final String TICKETS = "Tickets";
    static final String UPLOAD_TIME_KEY = "Upload-Time";
    private final LogSerializer mLogSerializer;
```

## II. Analyse dynamique :

### 1. SSL Pinning Bypass :

L'application **Bitwarden** utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques *MITM (Man In The Middle)* dans des canaux de communication sécurisés. Ainsi, pour pouvoir intercepter et analyser le trafic, il faut contourner le *SSL Pinning*.

Pour ce faire, nous allons suivre les mêmes étapes qu'on a fait pour contourner le *SSL Pinning* implémenté dans l'application LastPass.

#### 1) Télécharger le serveur Frida (*x86 Architecture*) :

```
[(kali㉿kali)-[~]]$ wget https://github.com/frida/frida/releases/download/16.0.7/frida-server-16.0.7-android-x86.xz
```

#### 2) Décompresser et renommer en '*frida-server*' :

```
[(kali㉿kali)-[~]]$ unxz frida-server-16.0.7-android-x86.xz
[(kali㉿kali)-[~]]$ mv frida-server-16.0.7-android-x86 frida-server
```

#### 3) Transférer le package '*frida-server*' à l'émulateur Android :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb push C:\Users\pc\Desktop\frida-server /data/local/tmp
C:\Users\pc\Desktop\frida-server: 1 file pushed, 0 skipped. 1.5 MB/s (53604060 bytes in 34.370s)
```

#### 4) Attribuer les permissions d'exécution au package :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell chmod 777 /data/local/tmp/frida-server
```

## App4 - Bitwarden

## 5) Démarrer le serveur Frida :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell /data/local/tmp/frida-server &
```

## **6) Obtenir le nom du package de l'application :**

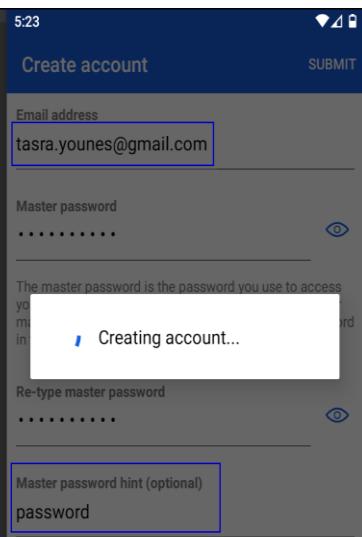
PID	Name	Identifier
4		
5466	Aptoide	cm.aptoide.pt
6009	Bitwarden	com.x8bit.bitwarden
5199	Email	com.android.email
5598	Keeper	com.callpod.android_apps.keeper
5275	Messaging	com.android.messaging
5629	NordPass	com.nordpass.android.app.password.manager
4915	Phone	com.android.dialer
5926	Settings	com.android.settings
-	Amaze	com.amaze.filemanager
-	Calendar	com.android.calendar
-	Camera	com.android.camera2
-	Clock	com.android.deskclock
-	Contacts	com.android.contacts
-	Custom Locale	com.android.customlocale2
-	Development Settings	com.android.development_settings
-	Files	com.android.documentsui
-	Gallery	com.android.gallery3d
-	LastPass	com.lastpass.lpandroid
-	Search	com.android.quicksearchbox
-	Superuser	com.genymotion.superuser
-	WebView Shell	org.chromium.webview_shell

## 7) Exécuter Objection pour contourner le *SSL Pinning* :

## App4 - Bitwarden

## 2. Interception du trafic :

## ▪ Signup :

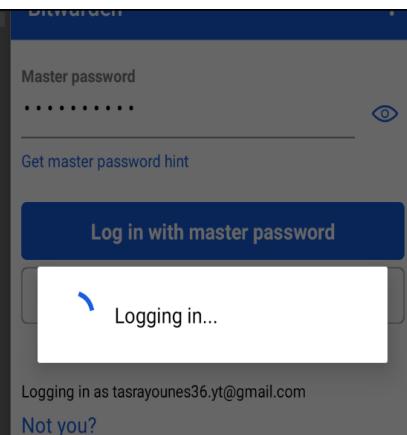


- On remarque que l'adresse email et le *Master Password Hint* sont envoyés en clair.
  - Le Master Password est envoyé sous forme de Hash.

#### ▪ Se connecter à Bitwarden

```
POST /connect/token HTTP/1.1
Content-Type : application/x-www-form-urlencoded
Accept : application/json
Auth-Email : dgFzcmF5h3VuZXMzN155dBBnbWFpbcb5jh20
Device-Type : []
Bitwarden-Client-Name : mobile
Bitwarden-Client-Version : 2022.11.0
User-Agent: Bitwarden_Mobile/2022.11.0 (Android 10; SDK 29; Model Nexus 4)
Accept-Encoding : gzip, deflate
Cookie : _cf_bm =
mfrHt0TX5astVQ3xK1qSHemTVYTxXmc0QhgFx_h4-1672873287-0-APHyPMyzGgzsMrxt509KdpysqDBqfsi5gAZ0c0UQ6c
QgkvJults2hly0ko3TfszoExFTXlRvCxLHpgf_QVvJw=
Content-Length : 262
Host : identity.bitwarden.com
Connection : close

scope=api+offline_access &client_id=mobile&grant_type=password &username=tamayounes16_yt4G@gmail.com
password=qwdAflikhucykh.y-1d4tF5pbeonfR3tHVOAxtSS5xkW03D &deviceType= &deviceIdentifier =
5834157a-4d7f-44b1-b0d0-0e99e3bb0b1c &deviceName=Nexus4 &devicePushToken =
```



- Lors de la connexion, le client est invité à saisir le Master Password.
  - On remarque que l'adresse email est envoyé en clair mais le Master Password est envoyé sous forme de nonce

## App4 - Bitwarden

### IV. Conclusion :

Properties		 bitwarden
Dangerous permissions		3
Unknown permissions		3
Encryption Standard		AES-256-CBC
Padding		PKCS7
Key Expansion Algorithm		PBKDF2 HMAC-SHA256
Secure Hashing Algorithms		✓
Secure PRG		✗
Zero Knowledge Architecture		✓
2FA Authentication		✓
MFA Authentication		✓
Biometrics		✓
Automatic form-filling		✓
Security alerts		✓
Mobile application		✓
Security audits		✗
Import of passwords		✓
Export of passwords		✓
SSO Authentication		✗
Password sharing		✓
Integrated database		✓
Certificate Pinning		✓
Root detection		✗
Encrypted traffic		✗
Encrypted email		✗
Encrypted Master Password		✓

## App 5 – 1Password:



Figure 19 : <https://1password.com/>

**1Password** est l'autre acteur majeur du marché des gestionnaires de mots de passe, et quoique différent sur le plan de l'interface, il possède de nombreuses similitudes avec **LastPass**. Les fonctions de création de rapports et d'intégration de **1Password** sont plus avancées. **1Password** bénéficie également de plusieurs partenariats avec des applications mobiles pour y intégrer directement ses fonctions, permettant aux utilisateurs de se connecter directement avec leur smartphone.

### I. Analyse statique :

#### 1. Permissions de l'application :

##### ▪ Permission critiques :

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

Cette permission permet à l'application de prendre des photos, des vidéos et de collecter les images que la caméra peut voir à tout moment.

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

Cette permission permet à l'application de lire à partir d'un stockage externe.

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

Cette permission permet à l'application de lire/modifier/supprimer le contenu du stockage externe.

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

Cette permission permet à l'application d'afficher des fenêtres d'alerte système. Les applications malveillantes peuvent prendre le contrôle de tout l'écran du téléphone.

##### ▪ Permission inconnue :

```
<uses-permission android:name="com.android.vending.BILLING"/>
```

##### ▪ Permissions normales :

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

Cette permission permet à l'application d'utiliser *Service.startForeground*

## App5 – 1Password

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

Ces permissions permettent à l'application de :

- ➔ Visualiser l'état du réseau.
- ➔ Créer des sockets réseau.

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

Cette permission permet à l'application d'interroger n'importe quelle application normale sur l'appareil, indépendamment des déclarations du manifeste.

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

Cette permission permet à l'application de se lancer dès que le système a fini le processus de démarrage. Cela peut rendre le démarrage du téléphone plus long et permettre à l'application de ralentir l'ensemble du téléphone en étant toujours en cours d'exécution.

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

Ces permissions permettent à l'application d'utiliser les modalités biométriques supportées par l'appareil.

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

Cette permission permet à l'application de contrôler le vibrateur.

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

Cette permission permet à l'application d'empêcher le téléphone de se mettre en veille.

### 2. Protection des données :

**1Password** est un fournisseur de sécurité ‘Zero-Knowledge’. En effet, le chiffrement se déroule sur les ordinateurs ou les smartphones avant d'être envoyé sur les serveurs du logiciel, ce qui signifie que le client est le seul à pouvoir déverrouiller le coffre-fort numérique de 1Password et accéder à ses informations.

**1Password** utilise un chiffrement de bout en bout avec un algorithme de chiffrement robuste **AES-256-GCM** avec **PBKDF2-HMAC-SHA256** pour la dérivation des clés, ainsi qu'une fonction de Padding **PKCS7**

```
public class EncryptionUtils {
    public static final int BLOCK_SIZE = 16;
    public static final String CIPHER_ALGORITHM_GCM_NOPADDING = "AES/GCM/NoPadding";
    public static final String CIPHER_ALGORITHM_NOPADDING = "AES/CBC/NoPadding";
    public static final String CIPHER_ALGORITHM_PADDING = "AES/CBC/PKCS7Padding";

    public class MyPBKDF2Engine {
        static final int AES_MAX_KEY_SIZE_BITS = 256;
        static final String KEY_ALGORITHM = "AES";
        static final byte[] SALT_CLAUSE = "Salted__".getBytes();
    }
}
```

## App5 - 1Password

### 3. Générateurs de nombres aléatoires non sécurisés :

L'application **1Password** utilise un générateur de nombres aléatoires non sécurisé `java.util.Random`

```
import java.util.Random;
import java.util.concurrent.TimeUnit;
import org.apache.commons.lang3.StringUtils;

/* Loaded from: classes.dex */
public final class DbxRequestUtil {
    private static final Random RAND = new Random();
    public static DbxGlobalCallbackFactory sharedCallbackFactory;

    /* Loaded from: classes.dex */
    public static abstract class RequestMaker<T, E extends Throwable> {
        public abstract T run() throws DbxException, Throwable;
    }
}
```

### 4. Divulgation des adresses IP :

L'application **1Password** divulgue des adresses IPv4 :

```
if (!z) {
    openDatagram = asyncServer.connectDatagram(new InetSocketAddress("8.8.8.8", 53));
} else {
    openDatagram = AsyncServer.getDefault().openDatagram(null, 0, true);
    Field declaredField = DatagramSocket.class.getDeclaredField("impl");
    declaredField.setAccessible(true);
    Object obj = declaredField.get(openDatagram.getSocket());
    Method declaredMethod = obj.getClass().getDeclaredMethod("join", InetAddress.class);
    declaredMethod.setAccessible(true);
    declaredMethod.invoke(obj, InetAddress.getByAddress("224.0.0.251"));
    ((DatagramSocket) openDatagram.getSocket()).setBroadcast(true);
}

if (!z) {
    openDatagram.write(new ByteBufferList(order));
} else {
    openDatagram.send(new InetSocketAddress("224.0.0.251", 5353), order);
}
return simpleFuture2;
ContentSigner build = new JcaContentSignerBuilder("SHA256WithRSA").build(keyPair.getPrivate());
JcaX509v3CertificateBuilder jcaX509v3CertificateBuilder = new JcaX509v3CertificateBuilder(x500Name, bigInteger, date, time, x500Name, keyPair.getPublic());
jcaX509v3CertificateBuilder.addExtension(new ASN1ObjectIdentifier("2.5.29.19"), true, new BasicConstraints(true));
return new JcaX509CertificateConverter().setProvider(bouncyCastleProvider).getCertificate(jcaX509v3CertificateBuilder.build(build));
```

### 5. Root detection :

L'application **1Password** implémente le *Root Detection* pour rendre l'exécution de l'application un peu plus difficile sur un appareil rooté.

## App5 - 1Password

```
public class Root {
    public static boolean isDeviceRooted() {
        return checkRootMethod1() || checkRootMethod2() || checkRootMethod3() || checkRootMethod4();
    }

    public static boolean checkRootMethod1() {
        String str = Build.TAGS;
        return str != null && str.contains("test-keys");
    }

    public static boolean checkRootMethod2() {
        try {
            return new File("/system/app/Superuser.apk").exists();
        } catch (Exception unused) {
            return false;
        }
    }

    public static boolean checkRootMethod3() {
        return new ExecShell().executeCommand(SHELL_CMD.check_su_binary) != null;
    }

    public static boolean checkRootMethod4() {
        return findBinary("su");
    }

    /* Loaded from: classes.dex */
    public enum SHELL_CMD {
        check_su_binary(new String[]{"./system/xbin/which", "su"});
        String[] command;
    }
}

DeviceInfo.root_status = String.valueOf(Root.isDeviceRooted());
DeviceInfo.isFilled = true;
```

### 6. Algorithmes cryptographiques non sécurisés :

L'application **1Password** utilise un algorithme cryptographique non sécurisé à savoir MD5, qui entraîne des collisions de hachage

Ces collisions signifient qu'un attaquant pourrait créer des fichiers qui auraient exactement le même hachage qu'un autre, ce qui rend impossible que le fichier n'a pas été falsifié.

```
public class FileCache {
    private static String hashAlgorithm = "MD5"; ←
    static MessageDigest messageDigest;

    static {
        try {
            messageDigest = MessageDigest.getInstance(MessageDigestAlgorithms.MD5);
        } catch (NoSuchAlgorithmException e) {
            MessageDigest findAlternativeMessageDigest = findAlternativeMessageDigest();
            messageDigest = findAlternativeMessageDigest;
            if (findAlternativeMessageDigest == null) {
                throw new RuntimeException(e);
            }
        }
    }
}
```

## App5 - 1Password

1Password utilise aussi l'algorithme de hachage SHA-1 qui entraîne aussi des collisions de hachage.

```
public static String sha1Hash(String str) throws B5EncryptionException {
    try {
        byte[] digest = MessageDigest.getInstance("SHA1").digest(str.getBytes());
        StringBuffer stringBuffer = new StringBuffer();
        for (byte b : digest) {
            stringBuffer.append(Integer.toHexString((b & UByte.MAX_VALUE) + 256).substring(1));
        }
        return stringBuffer.toString();
    } catch (NoSuchAlgorithmException e) {
        String str2 = "Error generating sha1 hash:" + Utils.getExceptionName(e);
        throw new B5EncryptionException(str2, str2);
    }
}
```

### 7. Certificate Pinning :

L'application 1Password utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques MITM (Man In The Middle) dans des canaux de communication sécurisés.

```
public void setTrustedCertificates(Collection<X509Certificate> collection) {
    try {
        TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance("PKIX");
        KeyStore keyStore = KeyStore.getInstance("jks");
        keyStore.load(null, null);
        int i = 0;
        for (X509Certificate x509Certificate : collection) {
            StringBuilder sb = new StringBuilder();
            sb.append("alias");
            int i2 = i + 1;
            sb.append(i);
            keyStore.setCertificateEntry(sb.toString(), x509Certificate);
            i = i2;
        }
        trustManagerFactory.init(keyStore);
        TrustManager[] trustManagers = trustManagerFactory.getTrustManagers();
        SSLContext sslContext = SSLContext.getInstance("TLS");
        sslContext.init(null, trustManagers, null);
        this.sslSocketFactory = sslContext.getSocketFactory();
    } catch (IOException | KeyManagementException | KeyStoreException | NoSuchAlgorithmException | CertificateException e) {
        throw new UncheckedIOException("Unable to initialize socket factory with custom trusted certificates.", e);
    }
}
```

### 8. Stockage externe :

L'application 1Password a les permissions de lecture/écriture sur le stockage externe, ce qui signifie que toute application malveillante installée sur le smartphone peut lire les données enregistrées sur le stockage externe.

```
public SyncResult doInBackground(Void... voidArr) {
    String parentDirectoryFromPath;
    File[].listFiles();
    if ("mounted".equals(Environment.getExternalStorageState())) {
        if (this.mPath.equals("/")) {
            parentDirectoryFromPath = Environment.getExternalStorageDirectory().getPath();
            this.mPath = parentDirectoryFromPath;
        } else if (this.mPath.equals(Environment.getExternalStorageDirectory().getPath())) {
            parentDirectoryFromPath = this.mPath;
        } else {
            parentDirectoryFromPath = Utils.getParentDirectoryFromPath(this.mPath);
            if (parentDirectoryFromPath == null) {
                parentDirectoryFromPath = Environment.getExternalStorageDirectory().getPath();
            }
        }
        File file = new File(this.mPath);
        ArrayList arrayList = new ArrayList();
        if (file.exists() && file.isDirectory() && !file.isHidden()) {
            for (File file2 : file.listFiles()) {
                if (file2.isDirectory() && !file2.isHidden()) {
                    arrayList.add(file2);
                }
            }
        }
    }
    return new FileBrowserResult(Enumerations.SyncStatusEnum.SUCCESS, this.mPath, parentDirectoryFromPath, arrayList);
}
```

## II. Analyse dynamique :

### 1. Bypass SSL Pinning :

L'application **1Password** utilise le *SSL Certificate Pinning* pour détecter et prévenir les attaques *MITM (Man In The Middle)* dans des canaux de communication sécurisés. Ainsi, pour pouvoir intercepter et analyser le trafic, il faut contourner ce mécanisme de sécurité.

Pour ce faire, nous allons suivre les mêmes étapes qu'on a fait pour contourner le *SSL Pinning* implémenté dans l'application Bitwarden.

#### 1) Télécharger le serveur Frida (*x86 Architecture*) :

```
(kali㉿kali)-[~]
└─$ wget https://github.com/frida/frida/releases/download/16.0.7/frida-server-16.0.7-android-x86.xz
```

#### 2) Décompresser et renommer en 'frida-server' :

```
(kali㉿kali)-[~]
└─$ unxz frida-server-16.0.7-android-x86.xz

(kali㉿kali)-[~]
└─$ mv frida-server-16.0.7-android-x86 frida-server
```

#### 3) Transférer le package 'frida-server' à l'émulateur Android :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb push C:\Users\pc\Desktop\frida-server /data/local/tmp
C:\Users\pc\Desktop\frida-server: 1 file pushed, 0 skipped. 1.5 MB/s (53604060 bytes in 34.370s)
```

#### 4) Attribuer les permissions d'exécution au package :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell chmod 777 /data/local/tmp/frida-server
```

#### 5) Démarrer le serveur Frida :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>adb shell /data/local/tmp/frida-server &
-
```

#### 6) Obtenir le nom du package de l'application :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools>frida-ps -Uai
  PID  Name                               Identifier
  4 -----
 3275  1Password                          com.agilebits.onepassword ←
 2768  Aptoide                            com.aptoide.pt
 2404  Email                             com.android.email
 2893  Keeper                            com.callpod.android_apps.keeper
 2494  Messaging                          com.android.messaging
 2944  NordPass                           com.nordpass.android.app.password.manager
 3157  Phone                             com.android.dialer
 2668  Settings                           com.android.settings
 2731  Superuser                          com.genymotion.superuser
      - Amaze                            com.amaze.filemanager
```

#### 7) Exécuter Objection pour contourner le SSL Pinning :

```
C:\Users\pc\Desktop\TP\MobileSec\platform-tools objection --gadget com.agilebits.onepassword explore  
Using USB device `Nexus 4`  
Agent injected and responds ok!  
  
[object] inject(ion) v1.11.0  
  
Runtime Mobile Exploration  
by: @leonjza from @sensepost  
  
[tab] for command suggestions  
com.agilebits.onepassword on (Google: 10) [usb] # android ssllibpinning disable  
(agent) Custom TrustManager ready, overriding SSLContext.init()  
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.verifyChain()  
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding TrustManagerImpl.checkTrustedRecursive()  
(agent) Registering job 318616. Type: android-ssllibpinning-disable  
com.agilebits.onepassword on (Google: 10) [usb] #
```

→ SSL Pinning est contourné avec succès

## 2. Interception du trafic :

## ▪ Signup :

```
POST /api/v1/signup HTTP/2
Host: start.ipassword.com
Content-Length : 171
Pragma: no-cache
Cache-Control : no-cache
User-Agent: Mozilla/5.0 (Linux; Android 10; Nexus 4 Build/QQD.200105.002; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/74.0.3729.186 Mobile Safari/537.36 lPasswordWebView/Android/7.9.3/70903004
Origin: https://start.ipassword.com
Accept-Language : en
Op-User-Agent : 1|A|70903003|22h7ezfhcxhwad4jk4iawidqla|B|1413|Chrome|74.0.3729.186|Android|10
Accept: /*
Accept-Encoding : gzip, deflate
X-Requested-With : com.agilebits.onepassword

{
    "email" : "tasrayounes36.yt@gmail.com",
    "domain" : "my",
    "name" : "tasrayounes36.yt@gmail.com",
    "type" : "",
    "language" : "en",
    "promocode" : "",
    "referrer" : "",
    "source" : "A",
    "tierName" : ""
}
```

# Let's get started!

Create your 1Password account  
today.

Your Email

tasrayounes36.yt@gmail.com

➔ Adress email est envoyé en clair

```
GET /api/v2/signup/385749 ?email=tasrayounes16.yt@gmail.com HTTP/2
Host: start.ipassword.com
Pragma: no-cache
Cache-Control: no-cache
Accept-Language: en
Op-User-Agent: Mozilla/5.0 (Linux; Android 10; Nexus 4 Build/QQID.200105.00Z; wv) AppleWebKit/537.36
User-Agent: Mozilla/5.0 (Linux; Android 10; Nexus 4 Build/QQID.200105.00Z; wv) AppleWebKit/537.36
Chrome/74.0.3729.186 Mobile Safari/537.36 iPasswordWebView/Android/7.9.3/70903004
Accept: */*
Accept-Encoding: gzip, deflate
X-Requested-With: com.agilebits.onepassword
```

**Check your email** for a 6-digit  
code

Enter your 6-digit code

385749



→ Saisie du code de vérification envoyé par l'application.

## App5 - 1Password

The image consists of two vertically stacked screenshots. The top screenshot shows a terminal or browser output of an HTTP request to 'txt/banlist/combined\_words.txt'. The response headers include 'Content-Type: text/plain' and 'Content-Length: 1307'. The body of the response is a list of approximately 1300 common password candidates, such as 'password', '123456', 'baseball', and 'jordan'. A red rectangular box highlights the first few lines of this list. The bottom screenshot shows a 'Sign up' page from 1Password. It features a 'Full Name' input field containing 'Hacker123', a checkbox for accepting terms and conditions, and a note about receiving emails. Both screenshots have a blue header bar with a back arrow and the text 'Sign up'.

```
GET /txt/banlist/combined_words.txt    HTTP/2
Host: app.1password.com
Pragma: no-cache
Cache-Control: no-cache
Origin: https://my.1password.com
User-Agent: Mozilla/5.0 (Linux; Android 10; Nexus 4 Build/QQID.200105.002; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/83.0.4103.109 Mobile Safari/537.36 1PasswordWebView/Android/7.9.3/70903004
Accept: /*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
X-Requested-With: com.agilebits.onepassword
```

HTTP/2 200 OK

```
Content-Type: text/plain
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, HEAD
Access-Control-Expose-Headers: Date
Access-Control-Max-Age: 3600
Last-Modified: Wed, 09 Nov 2022 16:10:28 GMT
X-Amz-Version-Id: VmQQUdDvhfTHCPMTTz6CkIZYK6G
Server: AmazonS3
Content-Security-Policy: default-src 'none';
Date: Thu, 05 Jan 2023 22:56:00 GMT
Cache-Control: max-age=31536000, public
Expires: Wed, 05 Jan 2033 15:15:56 GMT
ETag: W/"babbfdb8c895f25cc8fd2ff242e057c"
Vary: Accept-Encoding,Access-Control-Request-Headers,Access-Control-Request-Method
X-Cache: Hit from cloudfront
Via: 1.1 ae4d019557edbelef4b61effa4de6432.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: MAD50-C1
X-Amz-Cf-Id: M_CUEOHSH3WR8LsLloIPg6NCzQDoHEgCHN3GvAjm-DNDTfBqoiiI dpQ==
Age: 1307
```

password  
123456  
12345678  
1234  
qwerty  
12345  
dragon  
[REDACTED]  
baseball  
football  
letmein  
monkey  
696969  
abc123  
mustang  
michael  
shadow  
master  
jennifer  
111111  
2000  
jordan  
superman  
harley  
1234567

← Sign up

What should we call you?

Full Name

Hacker123

Get emails from 1Password with our latest announcements, product updates, advice, events, and research opportunities. [Unsubscribe](#) any time.

By proceeding, you agree to the [Terms of Service](#) and [Privacy Notice](#).

← Sign up

What should we call you?

Full Name

Hacker123

Get emails from 1Password with our latest announcements, product updates, advice, events, and research opportunities. [Unsubscribe](#) any time.

By proceeding, you agree to the [Terms of Service](#) and [Privacy Notice](#).

➔ Lors de la saisie du nom d'utilisateur, l'application compare le nom saisi avec une liste de noms interdits

## App5 - 1Password

### ■ Création du Master Password :

```
POST /api/v1/account HTTP/2
Host: my.1password.com
Content-Length : 5856
Pragma: no-cache
Cache-Control : no-cache
User-Agent : Mozilla/5.0 (Linux; Android 10; Nexus 4 Build/QQID.200104.1913.1413) AppleWebKit/537.36 (KHTML, like Gecko) 1Password Web View/Android/7.9.3/70903004
Origin: https://my.1password.com
Accept-Language : en
Op-User-Agent : 1A|70903004|igagsucqrtebfidgalaornkbm|B|1413|Chrome|74
Accept : */*
Accept-Encoding : gzip, deflate
X-Requested-With : com.agilebits.onepassword

(
  "signupUuid" :"VGK64GFETVCHEMBG4NIXXPKUYQ" ,
  "user": (
    "email" :"tasrayounes36.yt@gmail.com" ,
    "language" :"en" ,
    "name" :"Hacker123" ,
    "newsletterPrefs" :1,
    "accountKeyFormat" :"A3" ,
    "accountKeyUuid" :"573SNG" ,
    "avatar" :""
  ) ,
  "account" :(
    "name" :"tasrayounes36.yt@gmail.com" ,
    "domain" :"pg8-2bcnginweb4kgf2vxmdwdktzi" ,
    "type" :"I" ,
    "avatar" :""
  ) ,
  "device" :(
    "uid" :"igagsucqrtebfidgalaornkbm" ,
    "clientName" :"1Password for Web" ,
    "clientVersion" :"1413" ,
    "name" :"Chrome" ,
    "model" :"74.0.3729.186" ,
    "osName" :"Android" ,
    "osVersion" :"10" ,
    "userAgent" :
      "Mozilla/5.0 (Linux; Android 10; Nexus 4 Build/QQID.200105.002; wv) AppleWebKit/537.36 (KHTML, like Gecko) 1Password Web View/Android/7.9.3/70903004"
  )
),
```

➔ Le nom d'utilisateur et l'adresse email sont envoyés en clair.

Create your password.



• • • • • • •

SHOW



KEEP IT UP!

Remember: the longer and more random your password, the better!

Previous

Continue

```
"keyset": {
  "uuid" :"rquknvwglbb2san5ruww6rwf7u" ,
  "sn" :1,
  "encryptedBy" :"mp" ,
  "encSymKey" :{
    "kid" :"mp" ,
    "enc" :"A256GCM" ,
    "cty" :"b5+jwk+json" ,
    "iv" :"19cmzYjntoEdpsZ" ,
    "data" :
      "vsf0RXHu-cqSMT31SJWCvwqgbG8QuyF6PfgJZEQnA8iNHJZfou7wf1jUU7jaNvc9gwc
      s8wboOSUUAEBqJYJUDG_VRNZ-_3m5iHqjM56zBmpBH8g62fh7fm8MsUiCX4Wc"
    "alg" :"PBES2q-HS256" ,
    "p2c" :100000 ,
    "p2s" :"~kS646qDgWPNCuqkSZYg_A"
  } ,
  "pubKey" :{
    "alg" :"RSA-OAEP" ,
    "e" :"AQAB" ,
    "ext" :true ,
    "key_ops": [
      "encrypt"
    ] ,
    "kty" :"RSA" ,
    "n" :
      "tIkoedAFE05df5js6En6YMoBikKgOoUBpS4n1l2JdPqqtKPaq2pwUNAMjdwp_s3z
      ybiuXBwmjXm4Gno-nQwvZkU6T9AHGF9imOwsotto3fAPWcVqnZ6QEddogQF04awqor
      Ltw" ,
    "kid" :"rquknvwglbb2san5ruww6rwf7u"
  } ,
  "encPriKey" :{
    "kid" :"rquknvwglbb2san5ruww6rwf7u" ,
    "enc" :"A256GCM" ,
    "cty" :"b5+jwk+json" ,
    "iv" :"gk13e3oh3oot8eWY"
  }
},
```

Create your password.



• • • • • • •

SHOW



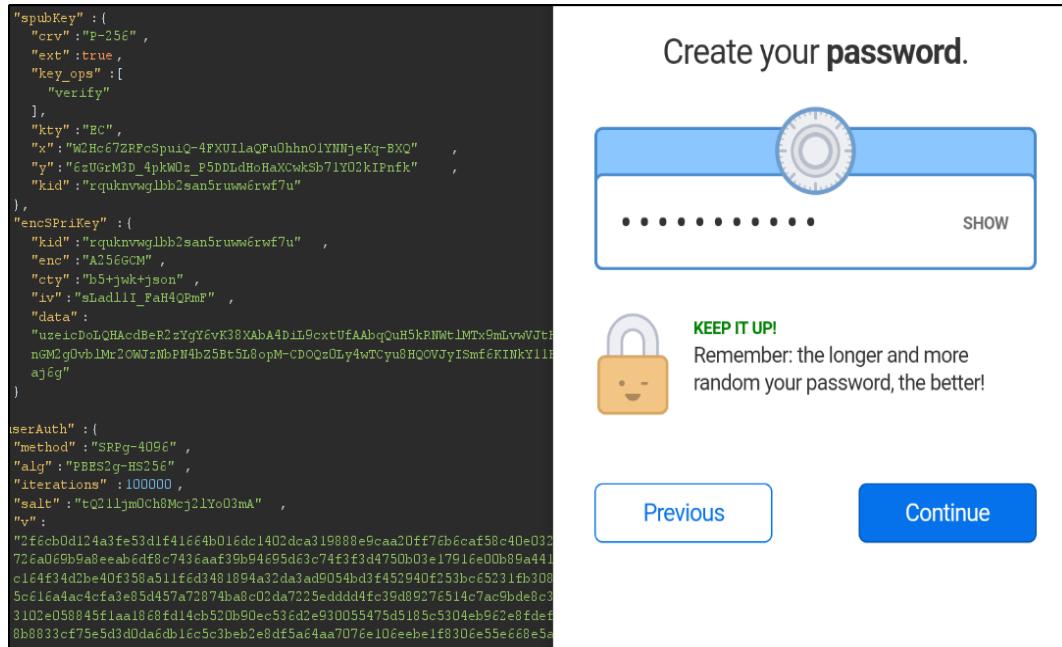
KEEP IT UP!

Remember: the longer and more random your password, the better!

Previous

Continue

## App5 – 1Password



➔ Envoi de paramètres cryptographiques chiffrés (Salt, keys, encrypted\_data, ...) nécessaires pour le chiffrement du Vault (Coffre-numérique)

### ▪ Se connecter à 1Password :

Après création d'un compte **1Password**, un email est envoyé contenant quelques instructions pour la connexion à 1Password, ainsi que la clé secrète (Secret Key) stockée dans un **Emergency Kit** qui est un document PDF contenant les détails du compte créé.



Figure 20 : sign-in details

Lors de la connexion **1Password**, le client est invité à saisir son adresse email, mot de passe ainsi que la clé secrète, qui est stockée dans le kit d'urgence (Emergency Kit). Afin de télécharger ce kit, le client doit d'abord se connecter sur le site Web de 1Password pour vérifier son adhésion puisque **1Password** est une application payante.

## App5 - 1Password

Cela signifie que, même si le client parvient à obtenir une version pro craquée de l'application **1Password**, il serait très difficile, voire impossible, de contourner le processus d'authentification et obtenir le kit d'urgence.

### III. Conclusion :

Properties	1Password
Dangerous permissions	4
Unknown permissions	1
Encryption Standard	AES-256-GCM
Key Expansion Algorithm	PBKDF2 HMAC-SHA256
Padding	PKCS7
Secure Hashing Algorithms	(✗)
Secure PRG	(✗)
Zero Knowledge Architecture	(✓)
2FA Authentication	(✓)
MFA Authentication	(✓)
Biometrics	(✓)
Automatic form-filling	(✓)
Security alerts	(✓)
Mobile application	(✓)
Security audits	(✗)
Import of passwords	(✓)
Export of passwords	(✓)
SSO Authentication	(✗)
Password sharing	(✓)
Integrated database	(✓)
Certificate Pinning	(✓)
Root detection	(✗)
Encrypted traffic	(✗)
Encrypted email	(✗)
Encrypted Master Password	(✓)

## Conclusion générale :

Les gestionnaires de mots de passe sont un outil important et de plus en plus nécessaire dans notre vie. Il est vrai qu'il n'existe pas de gestionnaire de mots de passe parfait et que chaque produit tente de trouver un équilibre entre la sécurité et la commodité pour les utilisateurs. A mon avis, le plus grand avantage des gestionnaires de mots de passe est d'aider les utilisateurs à stocker des mots de passe forts et aléatoires de sorte qu'ils puissent les utiliser pour différents services.

Le tableau ci-dessous synthétise les résultats de cette évaluation :

## Conclusion générale

Properties	 Keeper	 NordPass	 LastPas ...	 bitwarden	 1Password
Dangerous permissions	7	3	8	3	4
Unknown permissions	2	3	2	3	1
Encryption standard	AES-256-GCM	XChaCha20	AES-256-CBC	AES-256-CBC	AES-256-GCM
Key Expansion Algorithm	PBKDF2 HMAC-SHA256	_____	PBKDF2-SHA256	PBKDF2 HMAC-SHA256	PBKDF2 HMAC-SHA256
Padding	PKCS7	_____	PKCS7	PKCS7	PKCS7
Secure Hashing Algorithm	⊗	⊗	⊗	✓	⊗
Secure PRG	⊗	⊗	⊗	⊗	⊗
Zero Knowledge Architecture	✓	✓	✓	✓	✓
2FA Authentication	✓	✓	✓	✓	✓
MFA Authentication	✓	✓	✓	✓	✓
Automatic form-filling	✓	✓	✓	✓	✓
Security alerts	✓	✗	✓	✓	✓
Security Audits	✓	✗	✓	✗	✗
Biometrics	✓	✓	✓	✓	✓
Import of passwords	✓	✓	✓	✓	✓
Exports of passwords	✓	✓	✓	✓	✓
SSO Authentication	✓	✗	✓	✗	✗
Password sharing	✓	✓	✓	✓	✓
Integrated database	✓	✗	✓	✓	✓
Certificate pinning	✗	✓	✓	✓	✓
Root detection	✓	✓	✓	✗	✗
Encrypted traffic	✓	✗	✗	✗	✗
Encrypted email	✓	✗	✗	✗	✗
Encrypted Master Password	✓	✗	✓	✓	✓
<b>Security Score</b>	<b>4.8/5</b>	<b>4.3/5</b>	<b>4.5/5</b>	<b>4.4/5</b>	<b>4.8/5</b>

## **Conclusion générale**