ANDROID SECURITY PROJECT

CYBERSECURITE
ISEN TOULON

AGENDA

- Needed and Solution
  - Specification and constrains
  - Product presentation
- Focus on Security feature
  - Encryption
  - Keystore
  - File Integrity
  - Anti Tampering
  - Root detection
  - Anti Bidding
  - Offuscation
- Security Report (MOBsf)
- Enhancement forecast

SPECIFICATIONS

Functional needed

- Application based authentication
- GSM connexion not mandatory
- Data cyphering

Security feature

- Use of Android Keystore (key protection)
- Of-the-Fly encryption for medical data
- Specific encryption for sensible files
- Hacking protection capacity

LEGAL CONSTRAINS

References

- Rule n°2016/679 promulgated the apr 27th 2016 : EU General Data Protection Regulation
- French Public Health Code (art L-1111-8)
- French decree 2018-137 (feb. 26 2018)
- French law n° 78-17 (jan 6th 1978) (Loi Informatique et liberté)

Software limitation

- Limitation of data collected to what is strictly necessary
- Old files suppression
- Guarantee to third parties unauthorized third in particular to possible service providers
- ISO 27001, 27000 and 27018 certified host

Failure to comply with regulatory constraints constitutes an infringement of articles 226-16 226-17 of the French penal code and is punishable by 5 years' imprisonment and a € 300,000 fine

SOLUTION

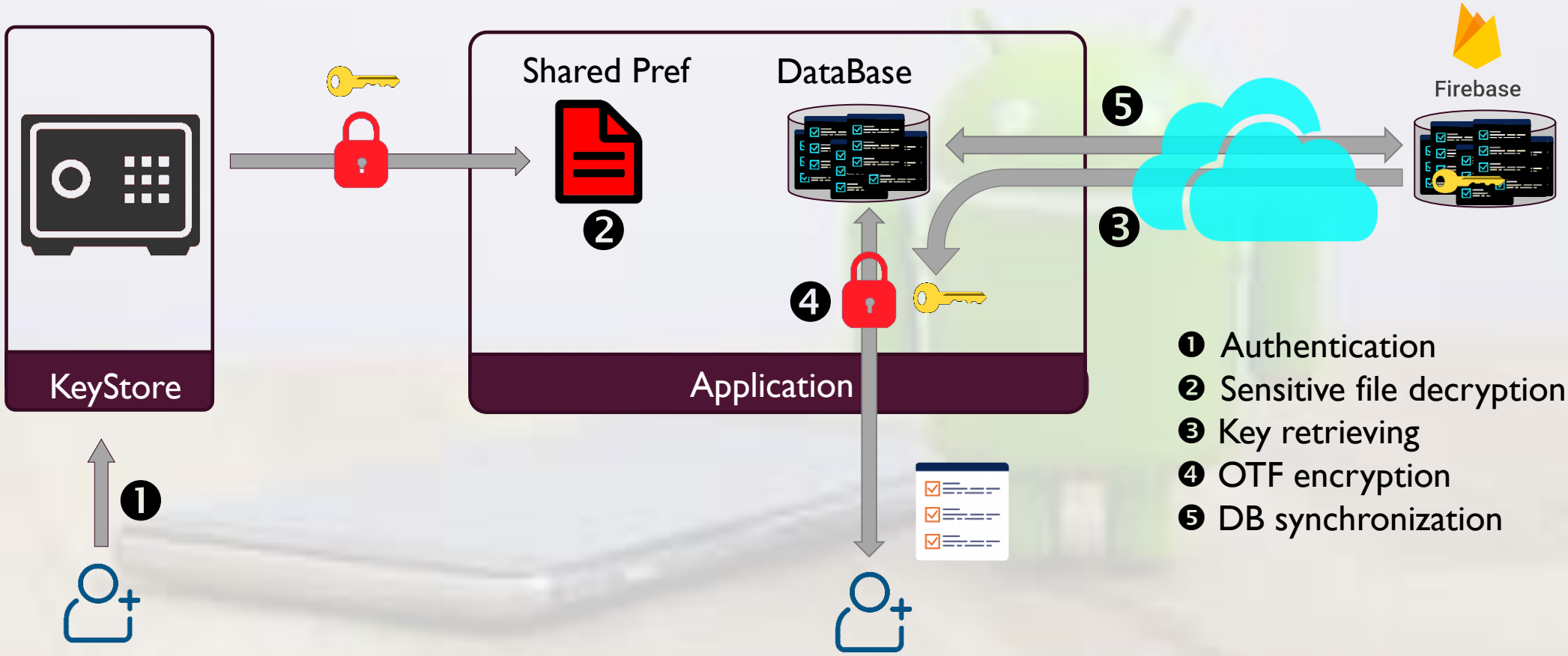
Technical choice

- Application
  - Restrain use of non native librairy
  - Restrain use of services
  - Use native functionality
- DB delagation to Firebase
  - Offline replication
  - 100% compatible with Android services

Security Benefits

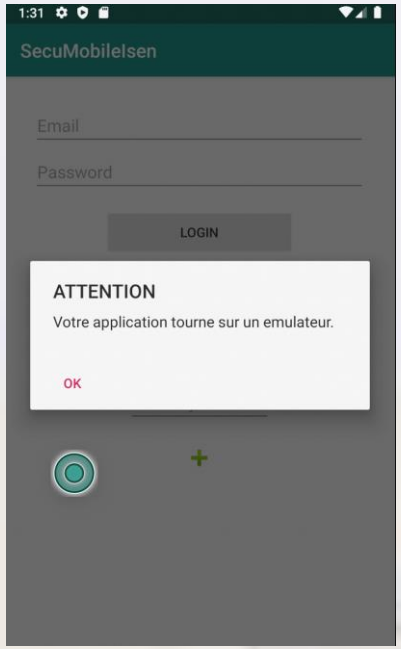
- Low surface exposure
- Restrict third party attack
- Application feature fully controlled
- Compliance with Fr and EU policy (FireBase : ISO 27001, 27000, 27018)

SOLUTION



❶ Authentication
❷ Sensitive file decryption
❸ Key retrieving
❹ OTF encryption
❺ DB synchronization

DATA SHEET



- Technical
  - Android API26 (Oréo)
  - API used
    - android.app *
    - android.content.Intent
    - android.os.Bundle
    - android.text.TextUtils
    - android.util.Log
    - android.view.View
    - android.widget *
    - androidx.appcompat.app.AppCompatActivity
    - kotlinx.android.synthetic.main.activity_form.*
    - java.security.KeyStore
    - androidx.recyclerview.widget.LinearLayoutManager
    - Java.util
    - javax.crypto.Cipher
    - com.google.firebase.auth.FirebaseAuth
    - com.google.firebase.database.DatabaseReference
    - com.google.firebase.database.FirebaseDatabase
  - Authorisation required
    - android.permission.ACCESS_NETWORK_STATE

- Security Feature
  - Integrity control
  - Key protection
  - OTF Encryptions
  - Hacking environment detection

DEMONSTRATION

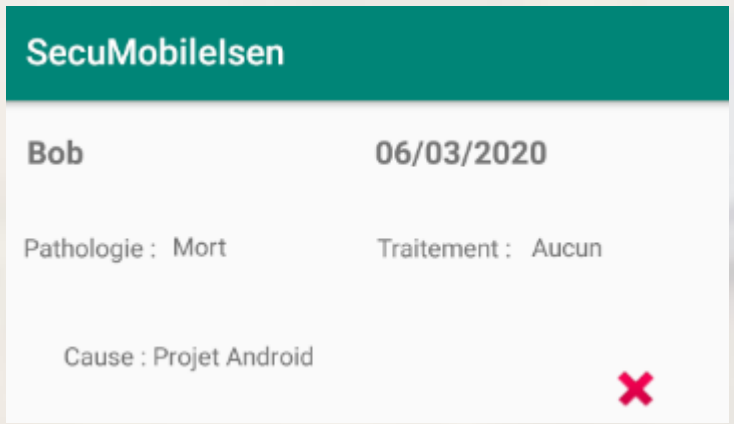
AGENDA

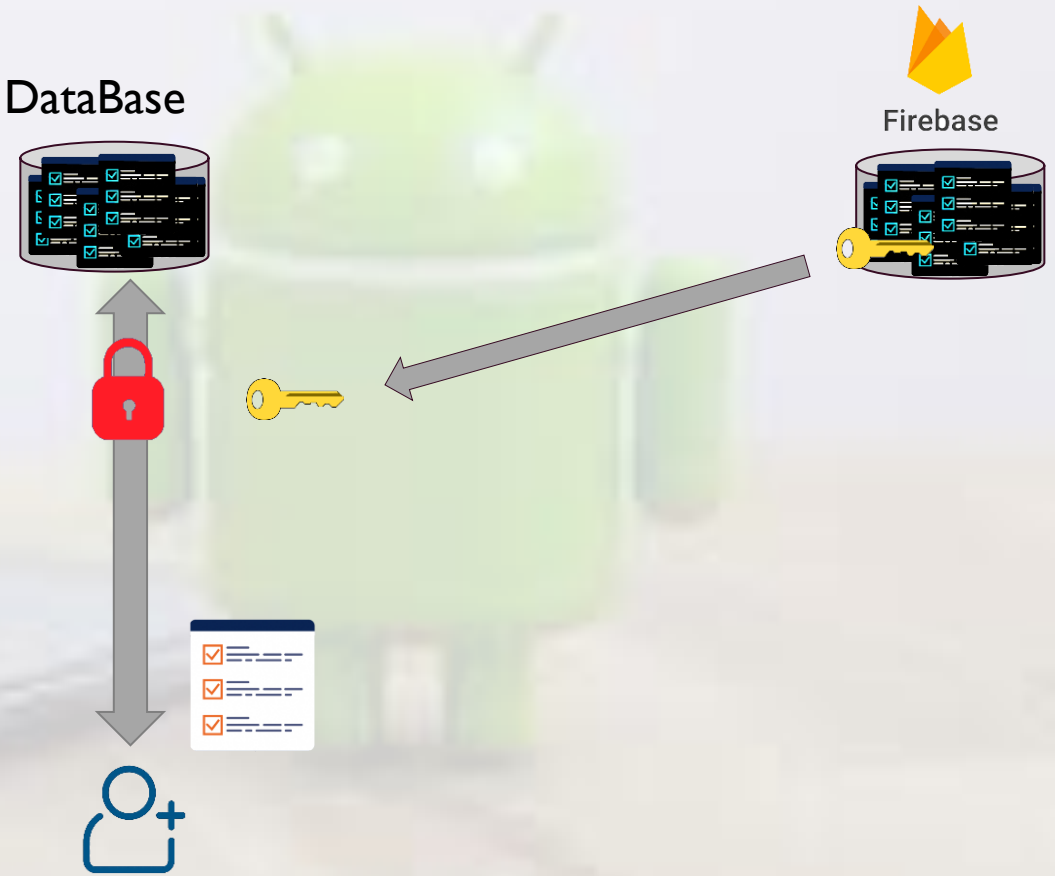
- Needed and Solution
  - Specification and constrains
  - Product presentation
- Focus on Security feature
  - Encryption
  - Keystore
  - File Integrity
  - Anti Tampering
  - Root detection
  - Anti Bidding
  - Offuscation
- Security Report (MOBsf)
- Enhancement forecast

ENCRYPTION

```
val docRef =
    db.collection( collectionPath: "masterKey").document( documentPath: "masterKey")
docRef.get().addOnCompleteListener { task ->
    if (task.isSuccessful) {
        val document = task.result
        if (document!!.exists()) {
val key = document.data.toString()
            Log.d( tag: "Key:" ,key)
val skeySpec = SecretKeySpec(key.toByteArray(), algorithm: "AES")

val cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS7Padding")
cipher.init(Cipher.ENCRYPT_MODE, skeySpec, IvParameterSpec)
cloudFirestore.collection( collectionPath: "patients")
    .document(newPatient.name.replace( oldValue: "/", newValue: "A"))
    .set(newPatient)
```





KEYSTORE

Technology

- Allow the app to save its own credentials that will be only accessible by the app
- A `KeyStore` manages different types of entries. Each type of entry implements the `KeyStore.Entry` interface. There are three kinds of entries:
  - KeyStore.PrivateKeyEntry
  - KeyStore.TrustedCertificateEntry
  - KeyStore.SecretKeyEntry

FILE INTEGRITY

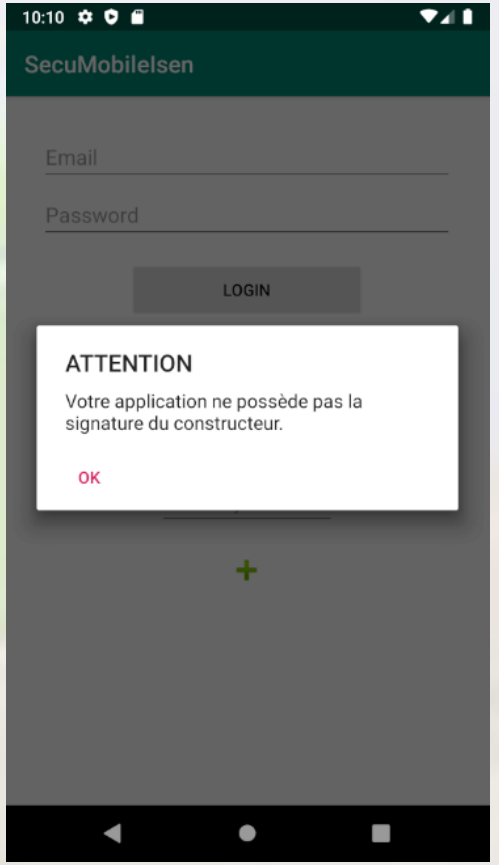
Signature

- The app must get its updates through the same provider
  - Storage of the developer signing certificate value hashed in SHA-512
  - Recalculation of the signature of the app at each start and comparison to the original

```
private fun getCurrentSignature(): String? {
    try {
        val packageInfo: PackageInfo =
            this.packageManager.getPackageInfo(this.packageName, PackageManager.GET_SIGNING_CERTIFICATES)

        for (signature in packageInfo.signingInfo.signingCertificateHistory) {
            val md: MessageDigest = MessageDigest.getInstance( algorithm: "SHA-512")
            md.update(signature.toByteArray())
            val currentSignature: String = Base64.encodeToString(md.digest(), Base64.DEFAULT)

            return currentSignature
        }
```



FILE INTEGRITY

Storage Protection

- The storage itself of the app should be checked
  - Use of BouncyCastle or SpongyCastle as Security provider
  - Production of an HMAC and protection in EncryptedSharedPreferences
  - Check of the integrity at each start

```
private fun initEncryptedSharedPreferences() {
    getSharedPreferences(preferencesName, MODE_PRIVATE).edit().apply()

    val masterKeyAlias = MasterKeys.getOrCreate(MasterKeys.AES256_GCM_SPEC)

    sharedPreferences = EncryptedSharedPreferences.create(
        preferencesName,
        masterKeyAlias,
        applicationContext,
        EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
        EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
    )
}
```


```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="AVdOT3s57o/rklfRW3XKuaMbWwGs+X9bw+98UmOopjK0TYVOQLI3T5VV">AR6t8pyvAUThEUo35IcpwQjlsLi6wLQlGTKiyp7G8TV+5Xt3XOKrZSRy7YCnkXjSuIbeeXV8BT9ZZQ==</string>
<string name="__androidx_security_crypto_encrypted_prefs_key_keyset__">12a9019ac40b3a7a07ecb4d4865dc46e1e8f415968204c4a4092ldc04e4a7af1a849996242f21098f9e542c4b6e46a754
<string name="__androidx_security_crypto_encrypted_prefs_value_keyset__">1288016c35de6f656b68bb2a0c1e587af11cb3e665a74988d2d7bb929fe1cbce84f588ea239f34f9d901edbd4f37b7d
<string name="AVdOT3t5DnLnTrFs6fbLwjo4b3Ueeg9EBiRu400VwT1qiV66Qg==">AR6t8pwqXrrcliMste0WkwYUgqprKx/lq70iqeXjMl7GeRQXyFYKFv+sJhvtslHf2Q==</string>
</map>
```

FILE INTEGRITY

Source Code

- The app should check if most of his files are unchanged
  - It often consist at a checksum or hash or these following files :
    - AndroidManifest.xml
    - Class files (*.dex)
    - Native Libraries (*.so)
  - We use generally Cyclic Redundancy Check (CRC)

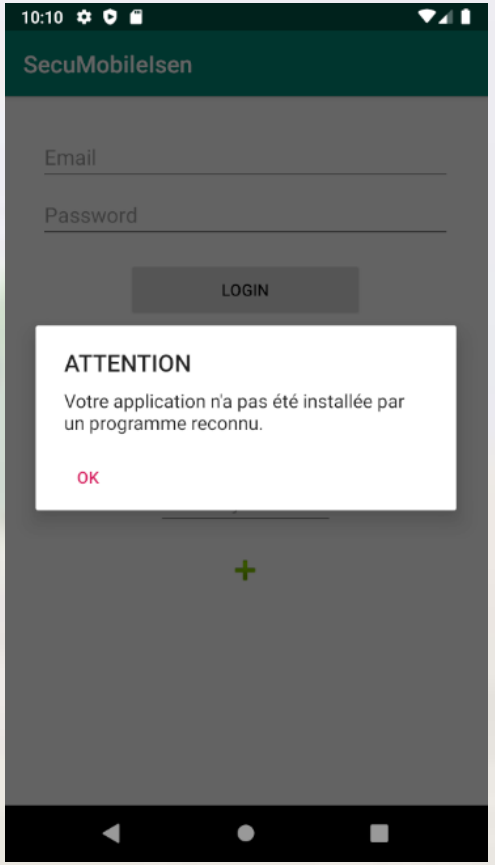
- In addition the AndroidManifest.xml hash verification allows protection against debugging

ANTI -TAMPERING

Installer verification

- The app must be installed from a valid organisation
  - We check the ID of the installer used for the app, like the one of the Google Play Store

```
private fun goodInstaller(): Boolean {
    val installer: String? = this.packageManager.getInstallerPackageName(this.packageName)
    return installer != null && installer.startsWith( prefix: "com.android.vending")
}
```

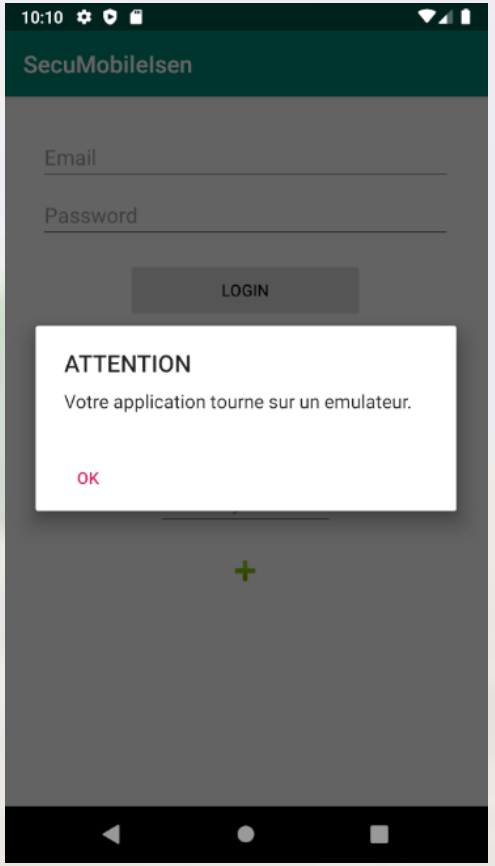


ANTI -TAMPERING

Emulator Detection

- Check of different values commons in emulators
  - Build.FINGERPRINT
  - Build.HARDWARE
  - Build.MODEL
  - Build.MANUFACTURER
  - Build.PRODUCT
  - …
- To extend to TelephonyManager files

```
(Build.BRAND.startsWith( prefix: "generic") && Build.DEVICE.star
 || Build.FINGERPRINT.startsWith( prefix: "generic")
 || Build.FINGERPRINT.startsWith( prefix: "unknown")
 || Build.HARDWARE.contains( other: "goldfish")
 || Build.HARDWARE.contains( other: "ranchu")
 || Build.MODEL.contains( other: "google_sdk")
 || Build.MODEL.contains( other: "Emulator")
 || Build.MODEL.contains( other: "Android SDK built for x86")
 || Build.MANUFACTURER.contains( other: "Genymotion")
 || Build.PRODUCT.contains( other: "sdk_google")
 || Build.PRODUCT.contains( other: "google_sdk")
 || Build.PRODUCT.contains( other: "sdk")
 || Build.PRODUCT.contains( other: "sdk_x86")
 || Build.PRODUCT.contains( other: "vbox86p")
 || Build.PRODUCT.contains( other: "emulator")
 || Build.PRODUCT.contains( other: "simulator"))
```



ANTI-ROOTING

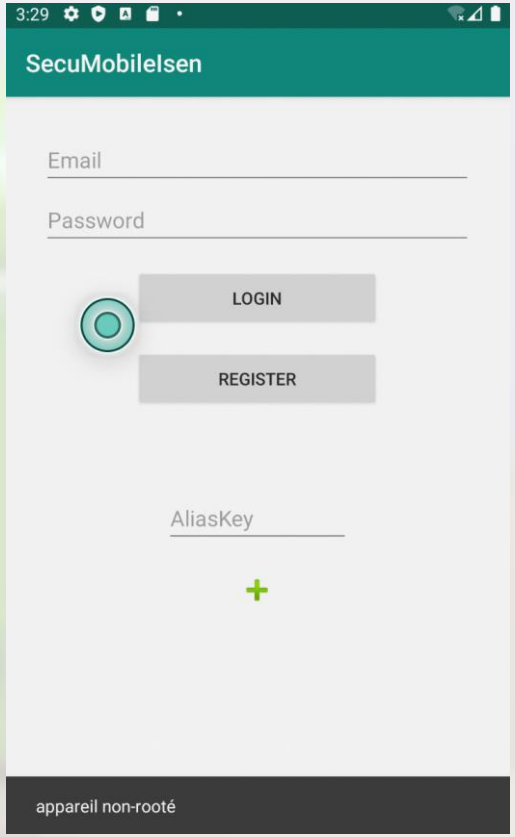
- What is root/rooted ?
- What is a rooted devices potentially dangerous to users/apps ?
- What is the interest to use a library for checking rooting ?

ANTI-ROOTING

```
private fun checkRooting(){

    var rootBeer  = RootBeer( context: this)

    if (rootBeer.isRooted) {
        val mySnackbar = Snackbar.make(mylayout, text: "appareil rooté", Snackbar.LENGTH_LONG)
        mySnackbar.show()
    }
    else {
        val mySnackbar2 = Snackbar.make(mylayout, text: "appareil non-rooté", Snackbar.LENGTH_LONG)
        mySnackbar2.show()
    }
}
```



ANTI-BINDING

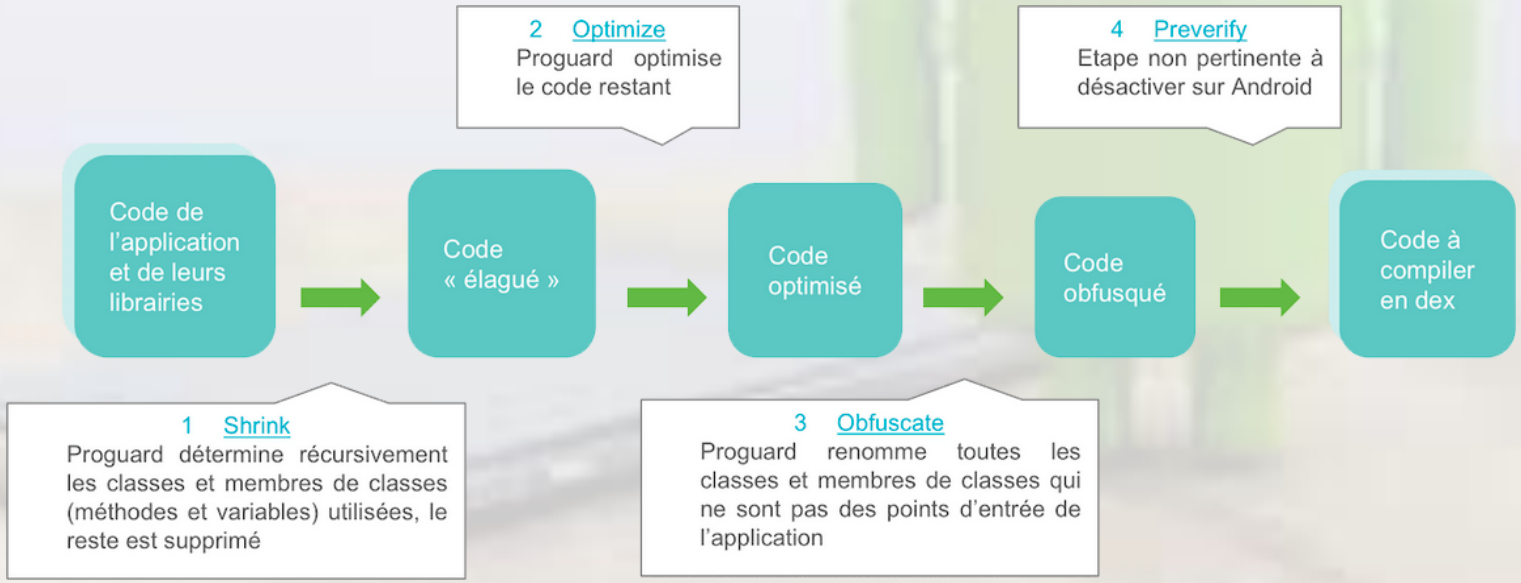
- What is binding ?

- Methods used for anti-binding:
  - Augmenting the credentials used for authentication
  - Encrypting the data stored in the device
  - Use token-based device authentication (Instance ID)

OBFUSCATION

- What is obfuscation ?

- ProGuard fonctionnement



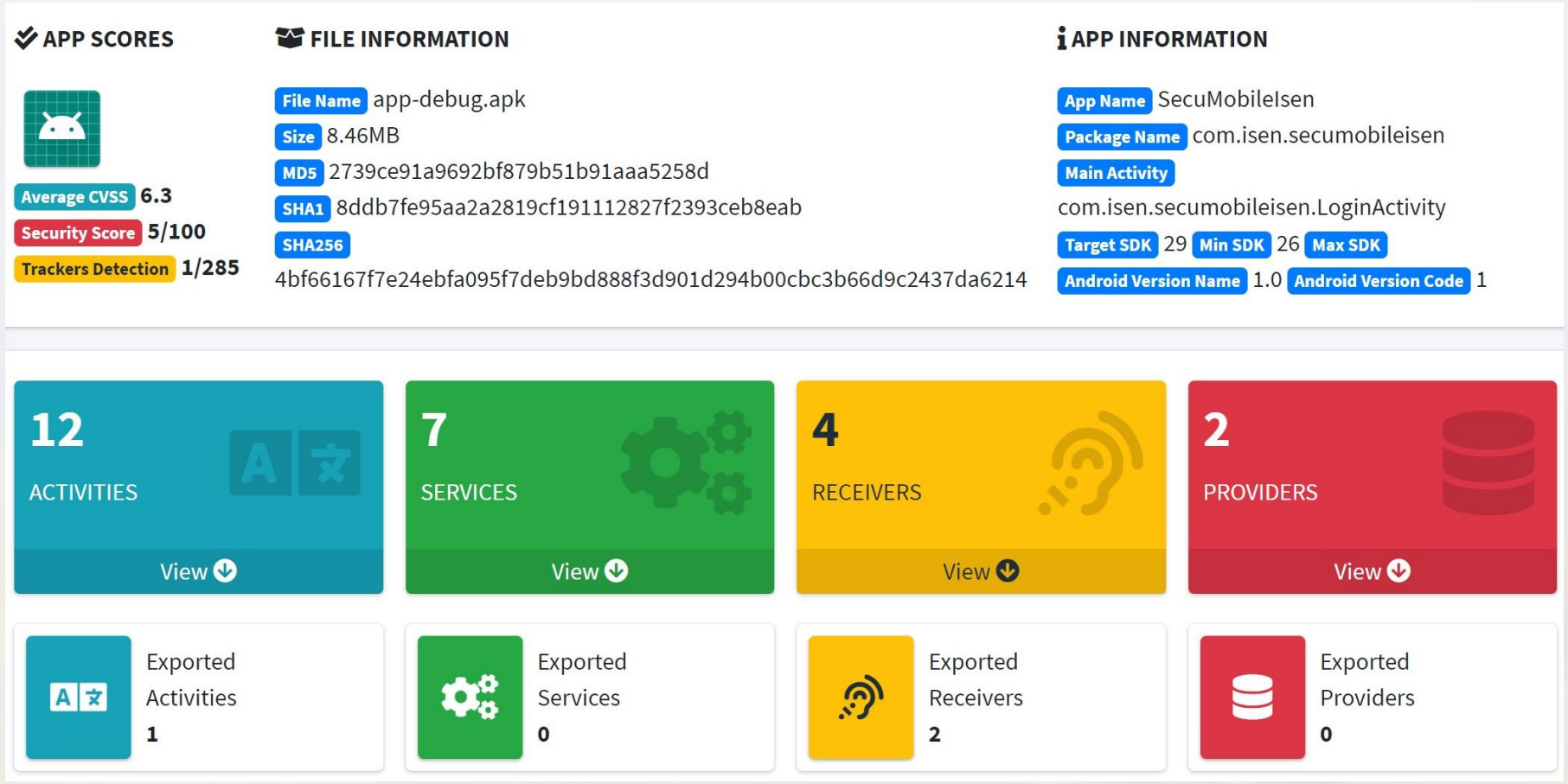
OBFUSCATION

- Result of obfuscation by ProGuard

AGENDA

- Needed and Solution
  - Specification and constrains
  - Product presentation
- Focus on Security feature
  - Encryption
  - Keystore
  - File Integrity
  - Anti Tampering
  - Root detection
  - Anti Bidding
  - Offuscation
- Security Report (MOBsf)
- Enhancement forecast

MOBsf REPORT



MOBsf REPORT

Services	Status	Explaination	Solution
android.permission.INTERNET	Dangerous	Mandatory	Accepted risk
android.permission.WAKE_LOCK	Dangerous	Mandatory (Firebase)	Accepted risk
com.google.android.finsky.permission	Dangerous	developer's information (Firebase)	Accepted risk
Permission	**Status**		
Activity (com.google.android.gms.measurement.AppMeasurementInstallReferrerReceiver)	High	permission set to Federated Sign In	Risk avoided
Broadcast Receiver (com.google.gsm.measurement.AppMeasurementInstall)	High	Permission set to INSTALL_PACKAGE	Further analysis to be completed
Broadcast Receiver (com.google.firebase.iid.FirebaseInstanceIdReceiver)	High	permission should be checked	Further analysis to be completed
Code Analysis	**CVSS**		
Files may contain hardcoded sensitive	7.4	Native functionality included in Util.java	Risk avoided with offuscation
MD5 / SHA-1 / Insecured Random	7.4	SHA1 was used to signe APK's certificat	
App uses ECB mode in Cryptographic	7.4		Switch to AES CBC mode
Rootbeer request root privileges	0	Native functionality included	Accepted risk

MOBsf REPORT

Services	Status	Explaination	Solution
android.permission.INTERNET	Dangerous	Mandatory	Accepted risk
android.permission.WAKE_LOCK	Dangerous	Mandatory (Firebase)	Accepted risk
com.google.android.finsky.permission	Dangerous		

Permission	Status
Activity (com.google.android.gms.measurement.AppMeasurementInstallReferrerReceiver)	High
Broadcast Receiver (com.google.gsm.measurement.AppMeasurementInstall)	High
Broadcast Receiver (com.google.firebase.iid.FirebaseInstanceIdReceiver)	High

Code Analysis	CVSS
Files may contain hardcoded sensitive	7.4
MD5 / SHA-1 / Insecured Random	7.4
App uses ECB mode in Cryptographic	7.4
Rootbeer request root privileges	0

Files may contain hardcoded sensitive informations like usernames, passwords, keys etc.	high	7.4	CWE-312	M9: Reverse Engineering	com/isen/secumobileisen/HistoActivity.java com/isen/secumobileisen/ListActivity.java io/grpc/internal/DnsNameResolver.java io/grpc/internal/ServiceConfigUtil.java io/grpc/internal/TransportFrameUtil.java io/reactivex/internal/schedulers/SchedulerPoolFactory.java io/opencensus/metrics/AutoValue_LabelKey.java io/opencensus/tags/AutoValue_Tag.java io/opencensus/trace/AutoValue_Tracestate_Entry.java	

Before

permission should be checked — Further analysis to be completed

Files may contain hardcoded sensitive informations like usernames, passwords, keys etc.	high	7.4	CWE-312	M9: Reverse Engineering	g/a/l1/t2.java

SHA1 was used to signe APK's certificat

Switch to AES CBC mode

Native functionality included — Accepted risk

MOBsf REPORT

SecuMobileIsen (1.0)

File Name:	app-release.apk
Package Name:	com.isen.secumobileisen
Average CVSS Score:	7.5
App Security Score:	100/100 (LOW RISK)
Trackers Detection:	1/285

MOBsf REPORT

Services	Status	Explaination	Solution
android.permission.ACCES_NETWORK_STATE	Dangerous	Mandatory	Accepted risk
Code Analysis	**CVSS**		
The App logs information	7,5	Native functionality included in FirestoreRecyclerAdapter.java FirestoreDataSource.java FirestorePagingAdapter.java	Accepted risk

AGENDA

- Needed and Solution
  - Specification and constrains
  - Product presentation
- Focus on Security feature
  - Encryption
  - Keystore
  - File Integrity
  - Anti Tampering
  - Root detection
  - Anti Bidding
  - Offuscation
- Security Report (MOBsf)
- Enhancement forecast

ENHANCEMENT FORCAST

- Encryption key rotation
- Authentication hardening (Factor 2 auth)
- Complete the file integrity check system
- Anti Frida

QUESTIONS ?