Android-Web challenge :

Rq : c’est une façon de faire, il est aussi possible de faire du mitm, de la redirection dns etc pour dumper les requêtes réseau de l’application, ou de l’analyse dynamique de l’apk via adb (compilé en debug) etc.

Part 1 : Reverse Android

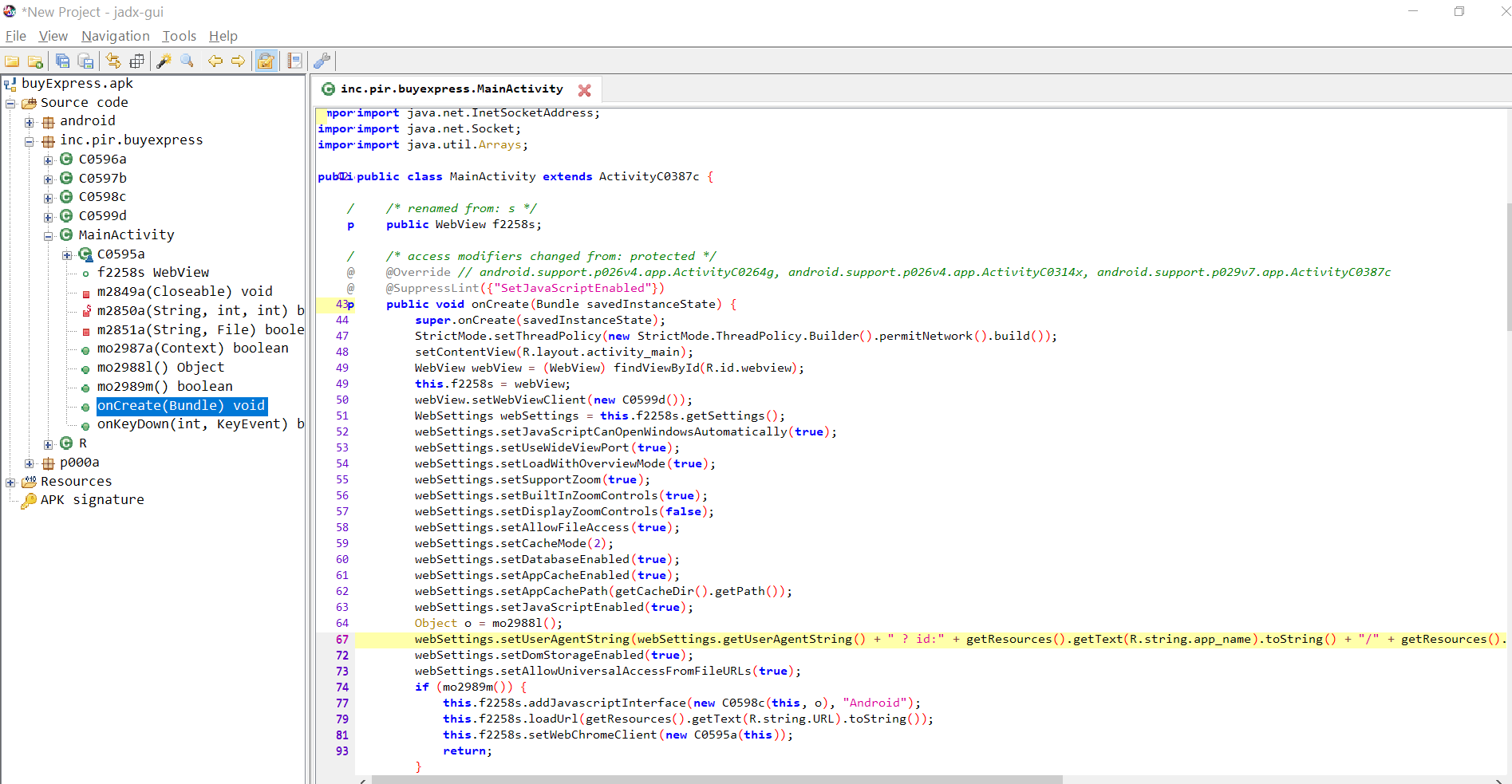
* 1. Fonctionnement général

L’application est une webapp qui se connecte à un site web et fourni quelques fonctionnalités native s(Showtoast). L’application comprend un apk chiffré qu’elle déchiffre au lancement, et qui fournit une méthode id() renvoyant un String, et qui est appelée depuis le site web. La sécurité de la connexion repose sur un user-agent personnalisé et ce String.

* 1. Récupération user-agent

Avec jadx par exemple :

webSettings.setUserAgentString(webSettings.getUserAgentString() + " ? id:" + getResources().getText(R.string.app\_name).toString() + "/" + getResources().getText(R.string.appVersion).toString() + "/" + getResources().getText(R.string.token).toString());



* 1. Récupération token

En utilisant apk-tool par exemple :

java -jar apktool\_2.3.4.jar -f d .\buyExpress.apk -o appcustom

cd appcustom/res/values

Regarder le fichier strings.xml :



On trouve le token à la fin et l’adresse du site internet

* 1. Déchiffrement et récupération id()

La fonction identification est importé depuis un apk chiffré :



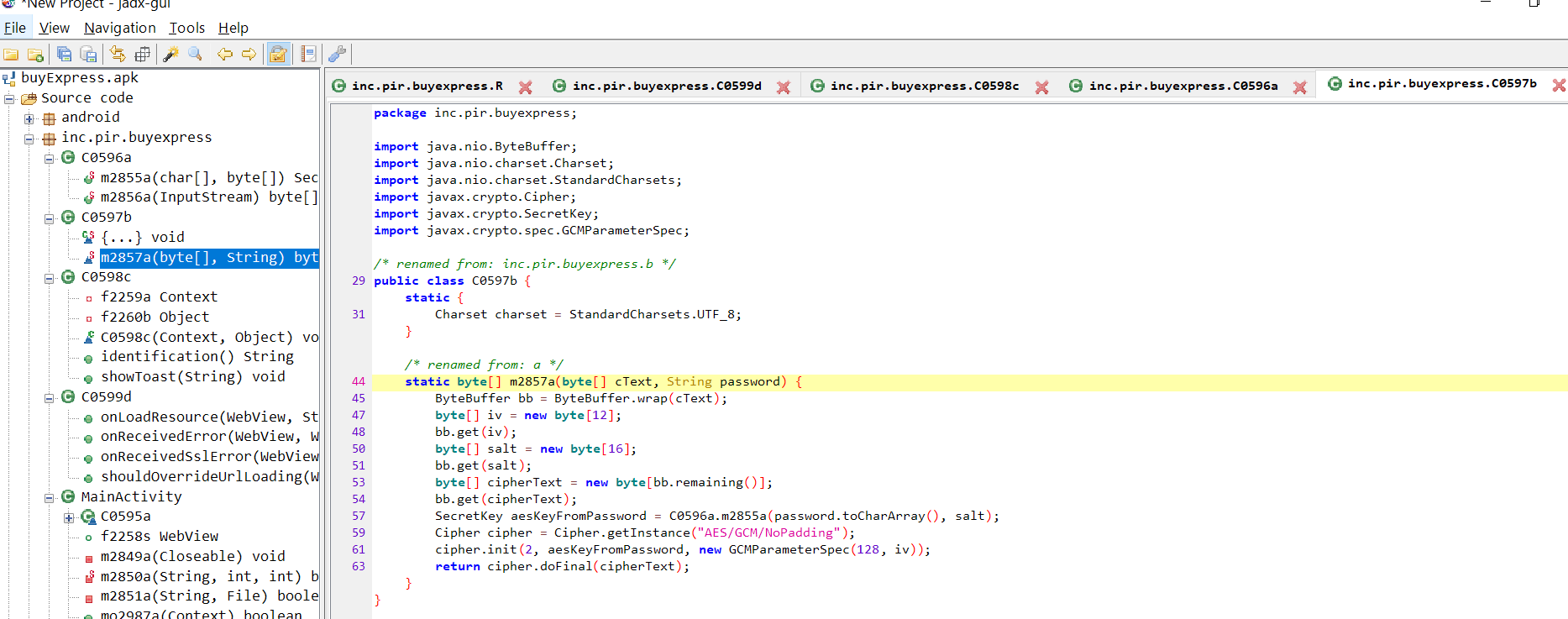


Elle sert à s’identifier.

En utilisant apk-tool :

Cd appcustom/assets

on trouve test-sec.apk qui est déchiffré dans le code java, comme vu avec jadx, avec le mot de passe « aBeautifulLayOfEncryption » et le chiffrement AES/GCM/NoPadding. On remarque que l’IV et le sel sont de 12 et 16 bytes en debut de fichier.





Le mot de passe est dérivé en utilisant PBKDF2WithHmacSHA256.

Un code java pour faire le déchiffrement est : 

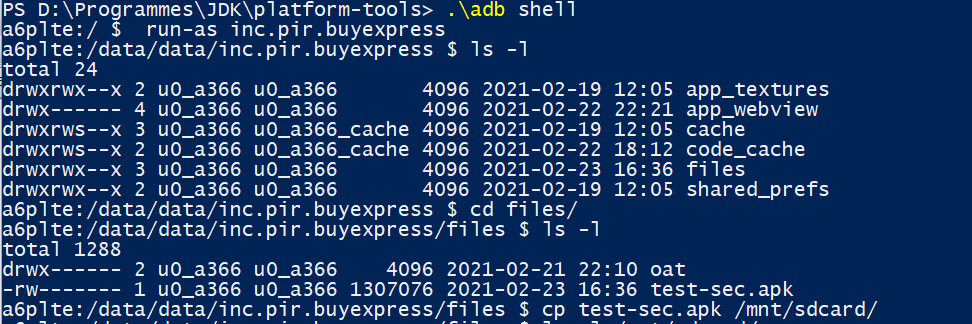
(Inspiré de https://mkyong.com/java/java-aes-encryption-and-decryption/)

javac .\EncryptorAesGcmPassword.java

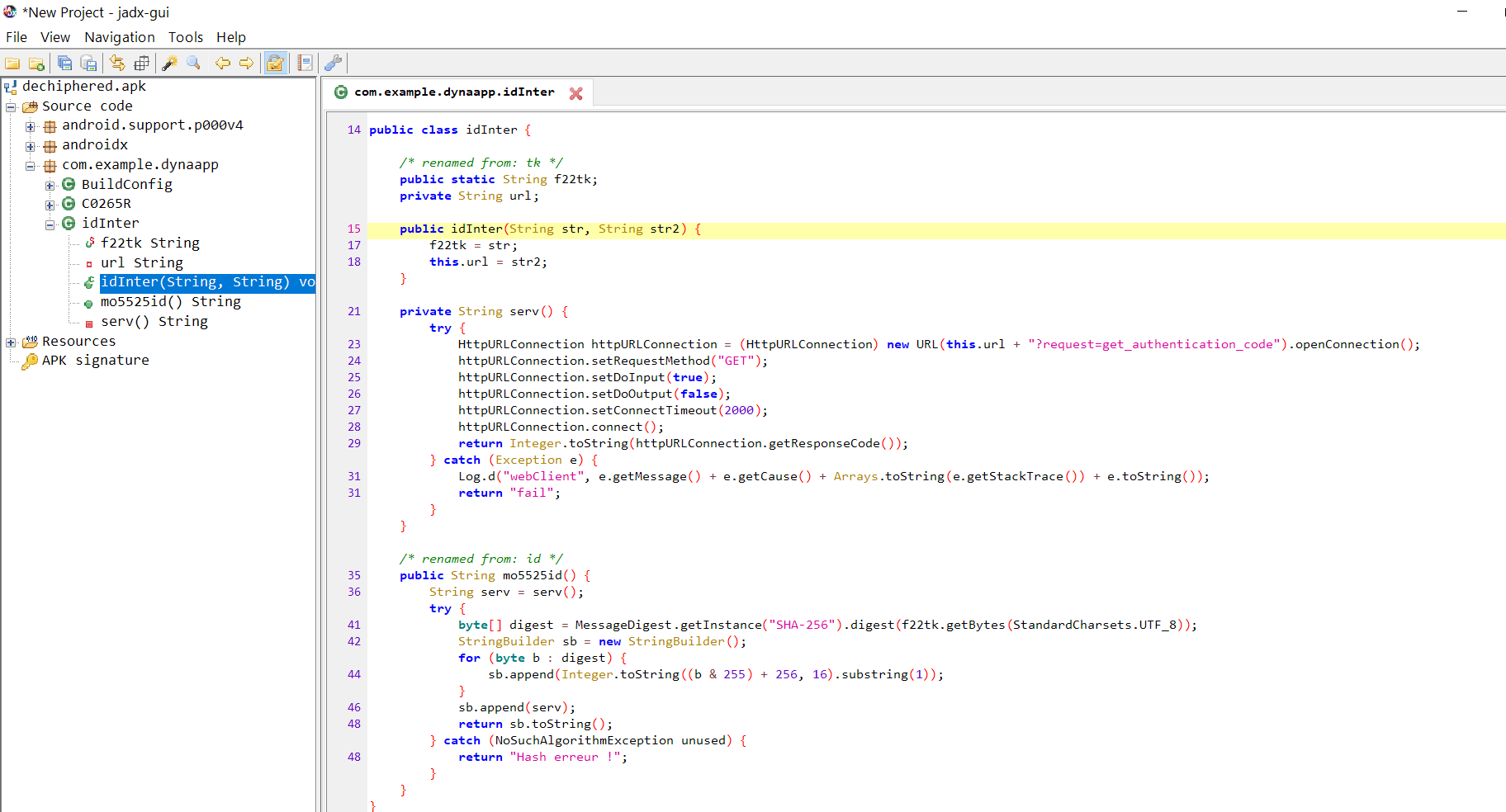
java -cp . EncryptorAesGcmPassword

Avec le fichier apk dans le même dossier.

ALT : obtenir le apk déjà déchiffré directement depuis le dossier de l’application (car fournie en mode debug)  
Rq : pointless security, as a rooted phone can get here on release apk too



Une fois le déchiffrement effectué, jadx nous livre les secrets de l’apk :

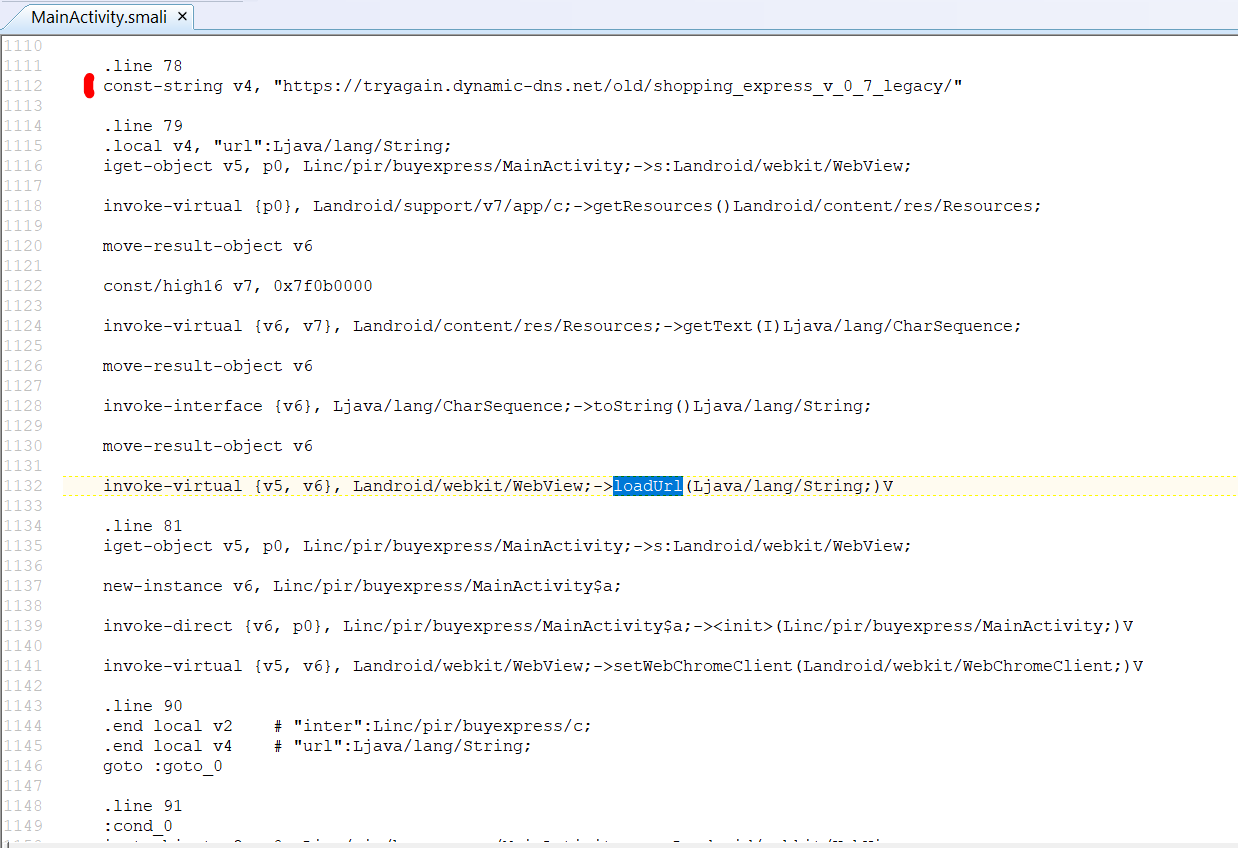


On comprend que le string d’authentification est en fait le sha256 du token déjà trouvé, auquel s’ajoute un code fourni par le code http du header de la réponse à la requête GET sur : site + **?request=get\_authentication\_code**

* 1. Récupération ancien site

En regardant dans le dossier smali (apk-tool) ou en utilisant ghidra :

Dans " appcustom\smali\inc\pir\buyexpress\MainActivity.smali"



Avant le chargement d’URL on peut trouver une ancienne URL oubliée dans le code.

Part 2 : Web

* 1. Connexion au site principal (ua+js) pc

On défini l’user agent de son navigateur comme il faut, par exemple :

Mozilla/5.0 (Android; Tablet; rv:19.0) Gecko/19.0 Firefox/19.0? id:BuyExpress/1.1/A1548S968D2563

(via une extension ou le about :config du navigateur)

On définit la fonction js : Android.identification comme suit :

class ndroid {

identification() {

alert("lol");

return "b69e8eff9a76f51ddad849f051127efe625e826c2e4e2bc4f45fe638ccf42616";

}

}

Android =new ndroid();

id=Android.identification();

(à compléter avec ajex serveur code http + virer code js useless).

* 1. Connexion à l’ancien site #diff #ua

Pareil mais version 0.7, visible dans les sources du site web.

* 1. Injection SQL

Login admin

Mdp :1’ OR « 1 »=’1

=>on est connecte

* 1. Découverte indice mot de passe

Nom des enfants + date de naissance, quelques essais, mot de passe : theophile0910

* 1. Remise du flag à 0

Curl (go jb !)

Enjoy !