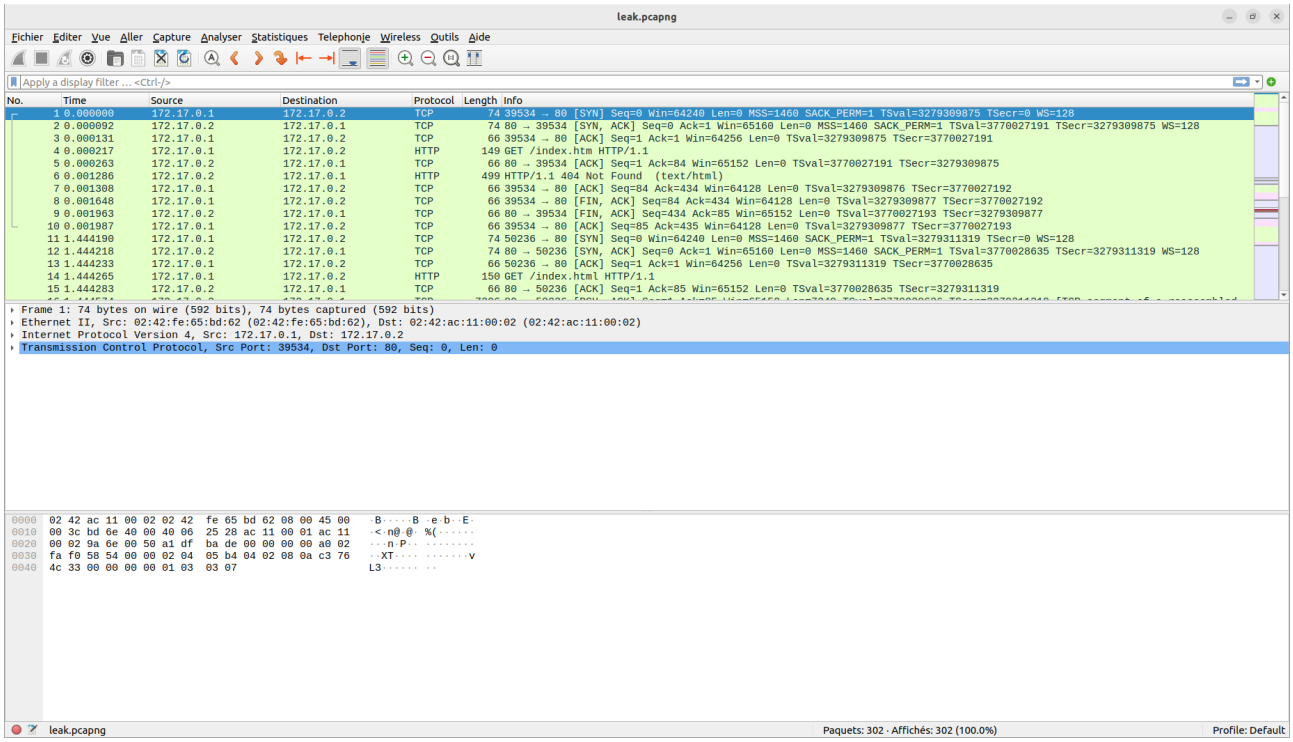
Write-up : Assassin’s communication

Dans ce document, je vais vous présenter une solution au challenge Assassin’s communication
Dans le cadre de ce challenge, il nous est fourni un fichier leak.pcapng.
Notre objectif ici est d’analyser cette capture réseau et de voir si on peut en tirer quelque chose.

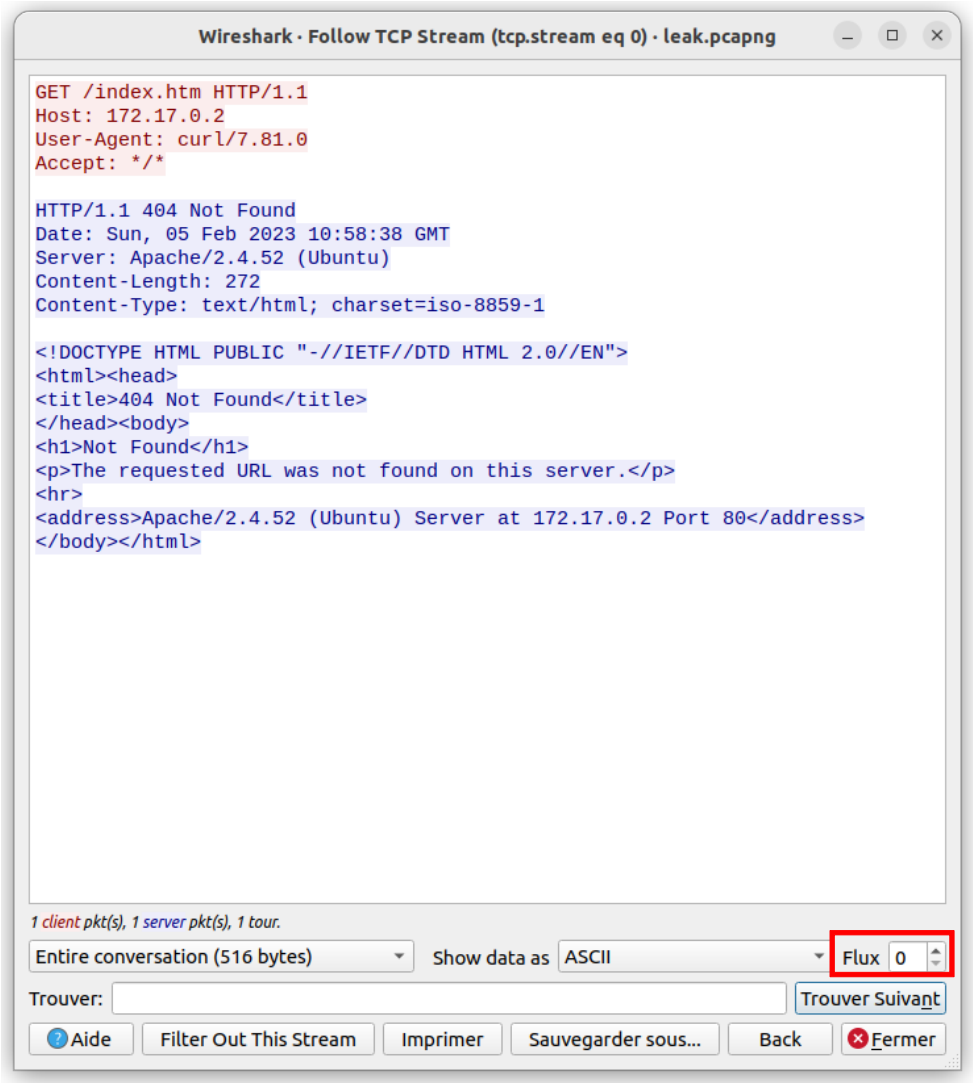
J’utilise ici wireshark qui est très pratique pour analyser des paquets réseaux facilement.

Ouvrons le fichier leak.pcapng avec wireshark :



Nous notons tout d’abord que ce fichier contient essentiellement une conversation entre deux postes réseaux d’adresses respectives 172.17.0.1 et 172.17.0.2.
Pour avoir une meilleure idée de l’échange entre les deux postes, nous pouvons suivre le flux TCP.

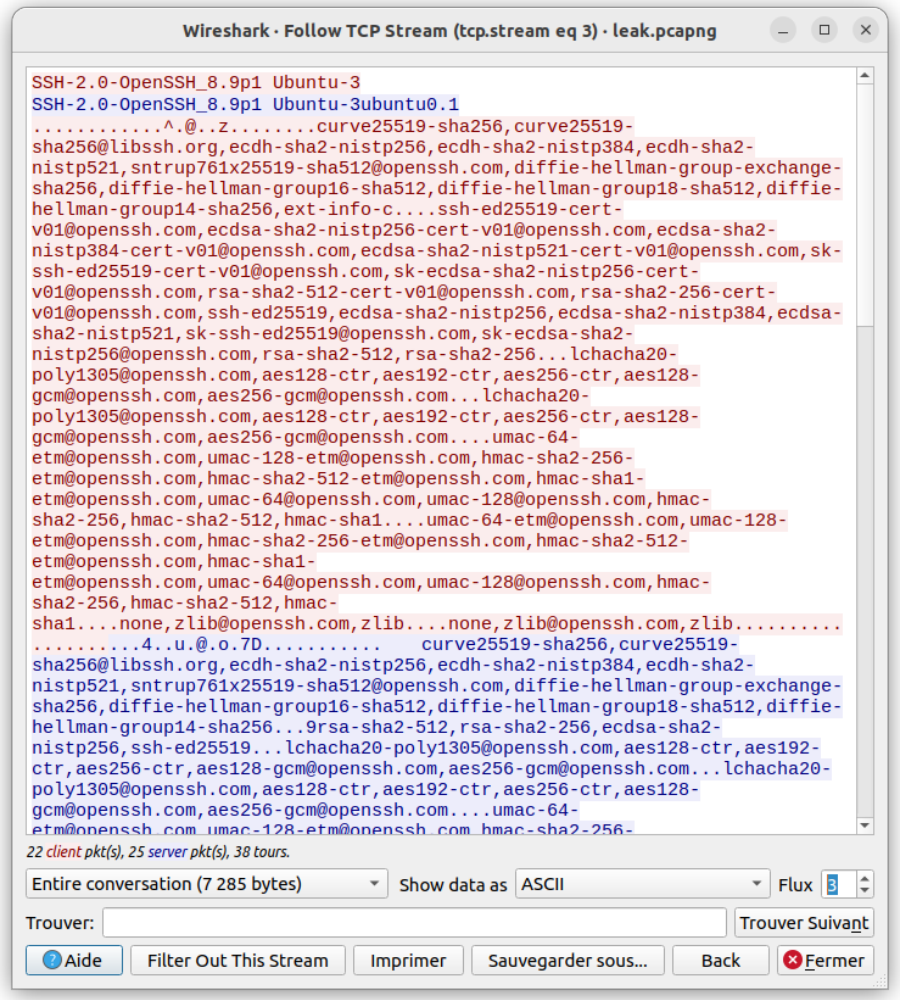
Pour cela, nous faisons un clic droit sur la première requête par exemple et nous allons dans « Suivre » puis « Flux TCP ».



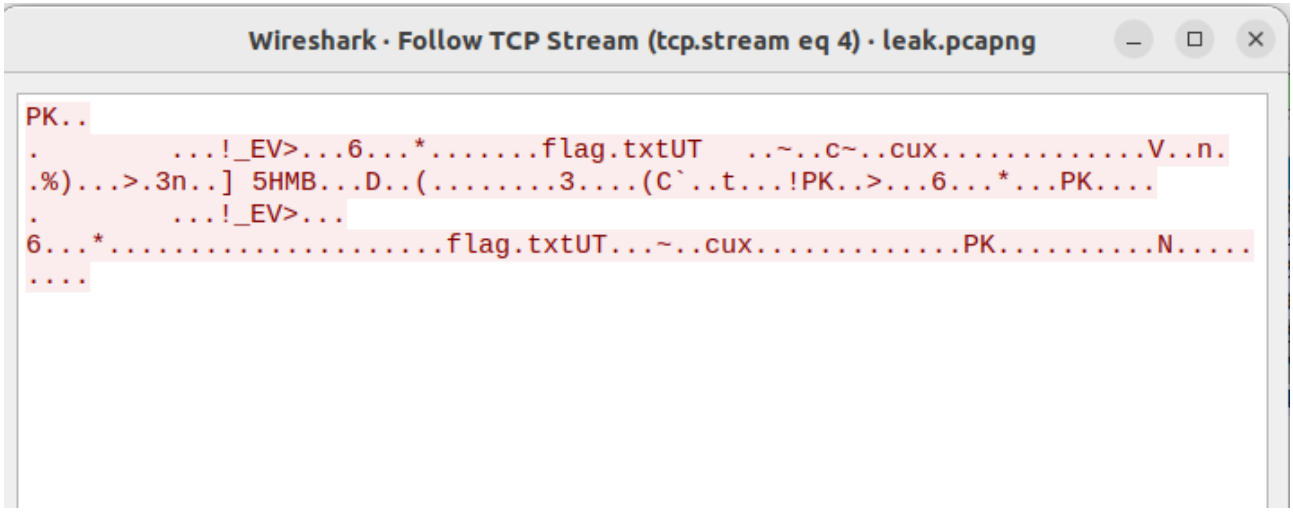
Maintenant que nous avons accès plus clairement à la conversation entre les deux, nous pouvons parcourir les différents flux TCP à l'aide de la liste déroulante en bas à droite.

En défilant les flux, nous notons qu'il s'agit principalement de requêtes GET, ici, la consultation d'une page web (par ex au flux 1, la page index.html fournit à l'installation d'un serveur apache2)

Au flux 3, nous notons une connexion à un serveur SSH.



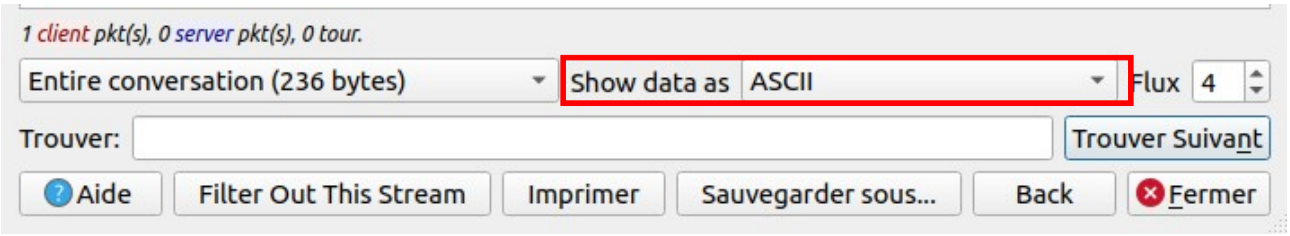
Si nous passons au flux suivant, le flux 4, nous notons quelque chose de très intéressant :



Ce qui nous attire l'oeil ici, est le « flag.txt ». Il s'agit certainement d'une transmission de donnée, ici plus précisément du fichier flag.txt (ce que nous voulons récupérer). Le « PK » au début du paquet, nous indique qu'il s'agit plus exactement d'un fichier zip. Nous en concluons donc qu'un poste envoie un fichier zip à l'autre poste contenant le flag que nous voulons récupérer.

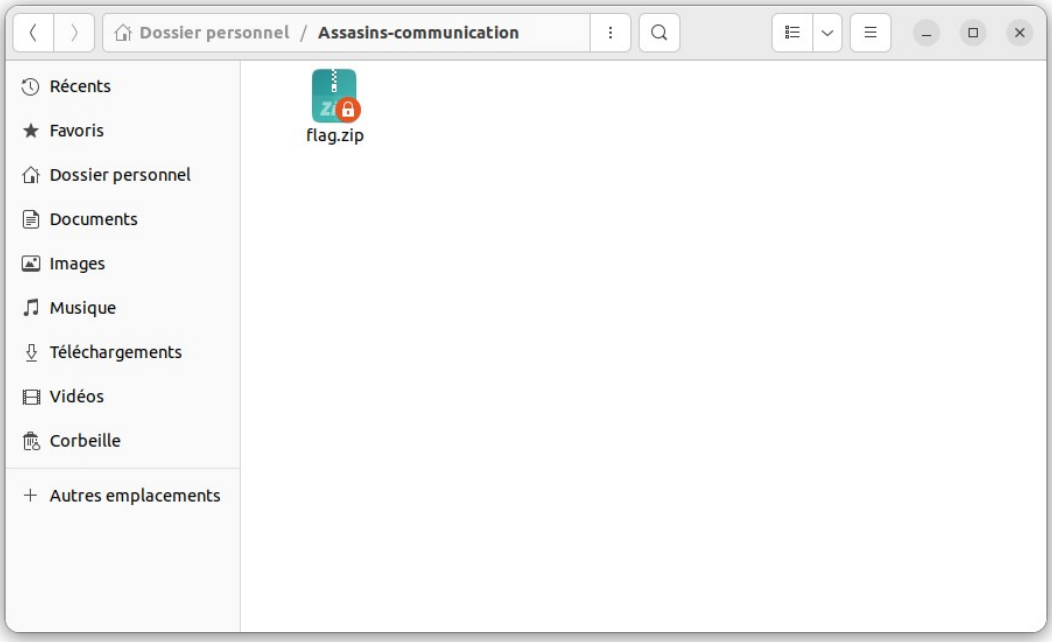
Notre première objectif est de récupérer ce fichier zip. Wireshark nous permet de faire ça assez facilement.

En bas de la page, nous avons la possibilité de choisir le type d'affichage des données.



Nous choisissons d'afficher les données en « Brut » pour après télécharger un fichier de données de base. Le fait de télécharger les données de manière brute rend le format de donnée universelle qui dans tout les cas sera compris peut importe l'extension du fichier.

Après avoir passé les données en brute, nous pouvons sauvegarder sous les données dans un fichier zip.



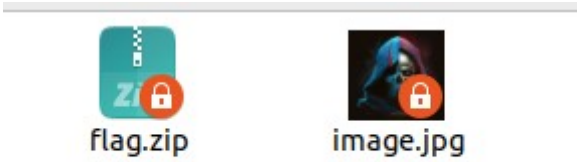
Hélas, quand nous essayons d'extraire le zip, un mot de passe est requis.

Nous allons devoir fouiller un peu plus dans la capture réseau.

Revenons aux flux TCP.
En passant différents flux contenant peu d'informations intéressantes, nous nous arrêtons finalement au flux 8. Cela nous paraît être encore ici une transmission de donnée, et grâce au préfixe « JFIF » nous pouvons deviner qu'il s'agit d'une image jpeg.

```
.....JFIF.............C.........................
        ......................"..."*%%*424DD\...C.......................
                                ......................"..."*%
%*424DD\......".3.."....................................
.....................}........!1A..Qa."q.2....#B...R..$3br.
.....
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz.............................
..............................................................................
.....................w.......!1..AQ.aq."2...B....     #3R..br.
.$4.
%.....&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz.........................
.....................................................?..p}.....tR..@
QE% ..JZ.(...))h...E..QK.(.(........LR....v.w...j].o.
n..............K.....T...e....;q.{..7..(..e.H...M......<....
1."...SQ...W.......A...R.....Yx....y.U....v.i.8..K...
3i=..I4F2.h....N;.....,.q...J.ZJ(..(.Q@
...p.1h..@....(.....
(.....(..f.......zP..V.Cjb,.C(......:g..A.'u.
..j.B3NE..z
.`......W-...f.q.)^.(...~........fL`..y8...M_Vx;.o.4B/5...r.U.Ku5...bqZ.
C.....G.eI..h..)..G.....E.I....qr..!.>.^....^!..M........?
9.......X4.h`..v...]..To..h..K..........N./9u...8.;.W....B....g.^/....+..
6K[.-!r!....Tn.\RBX....._... .$...........g....C....(F...X.0.f..g..
3_5...u.R..m6..Aex|E|.-.U...0.d.vz.3...
{..;r..QePN........HZ.Ge.,.e..|...y+...X[d.U.$..r../V<u...V.......s$@..
```

Nous allons vouloir ici aussi récupérer l’image transmise. Pour ce faire, nous allons suivre la même procédure en passant les données en brute et sauvegarder sous les données brutes dans une image.

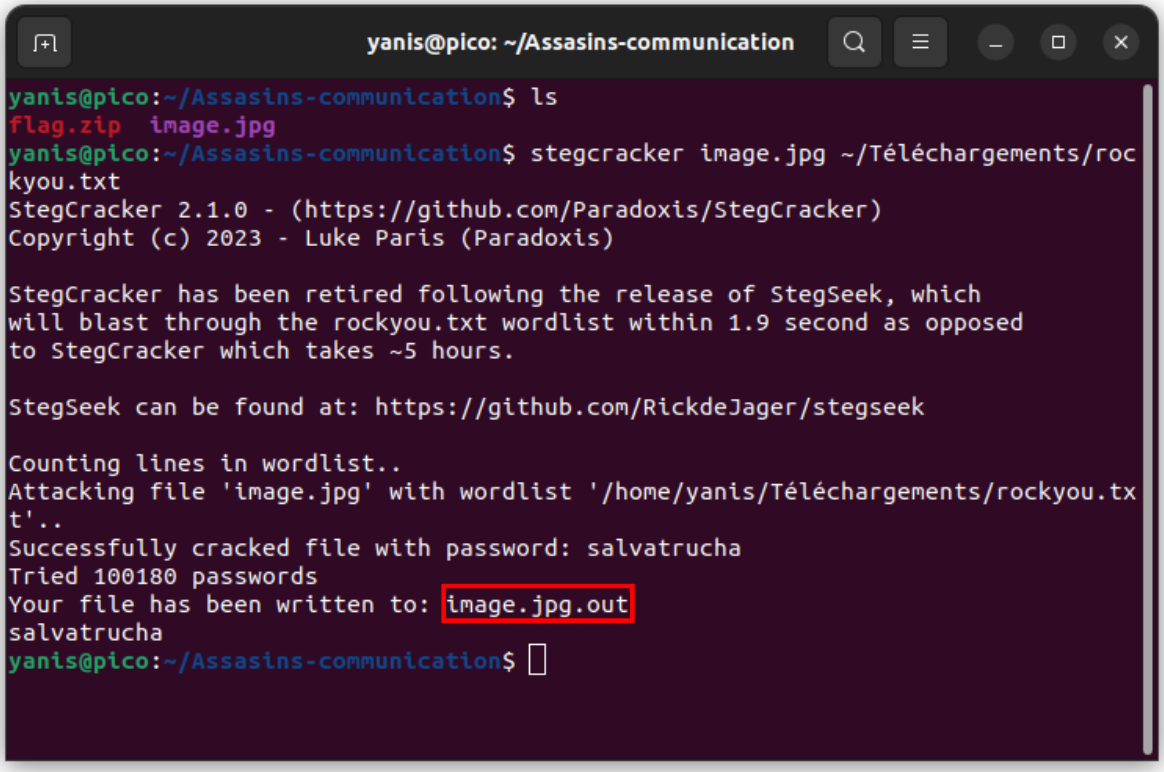


L’image que nous avons extraite est finalement celle ci :

En parcourant le reste des flux, nous ne tombons sur plus rien de vraiment intéressant.
Nous aurions donc tout ce qu’il nous faut pour retrouver le flag à l’aide de ces deux fichiers.
Rappelons que le fichier zip est protégé par un mot de passe. Ce que nous pouvons tenter ici, est de vérifier si l’image ne serait pas une couverture cachant des données qui pourrait nous intéresser, par exemple le mot de passe du fichier zip. Cela pourrait permettre aux deux de transférer des données de manière discrètes.

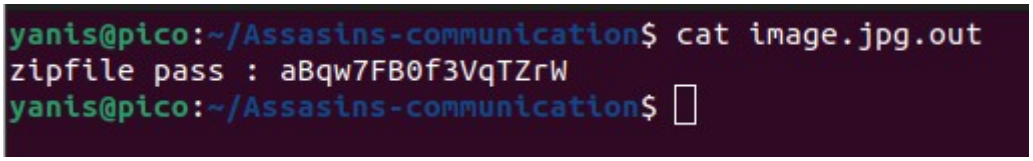
Nous utiliserons ici, l’outil stegcracker et la wordlist rockyou.txt pour tenter de récupérer les potentielles données cachées dans l’image.

Allons-y :

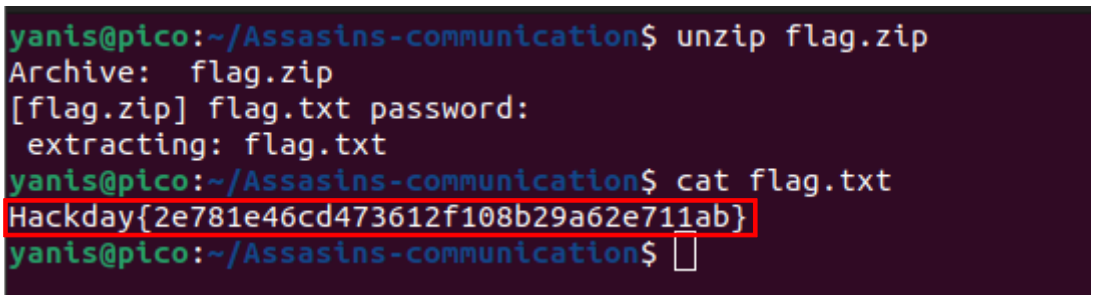


Nous avons finalement réussi à extraire des données de l'image récupérer de la capture réseau.
On nous précise que ces données ont été écrites dans le fichier image.jpg.out et que le mot de passe utilisé pour cacher les données était « salvatrucha »

Lisons le contenu de ce fichier « image.jpg.out »



Nous obtenons le mot de passe du fichier zip.
Il ne nous reste plus qu'à l'extraire avec ce mot de passe et récupérer le flag.



Nous avons finalement réussi à récupérer le flag ce qui clôture avec succès ce challenge.

Auteur : Yam