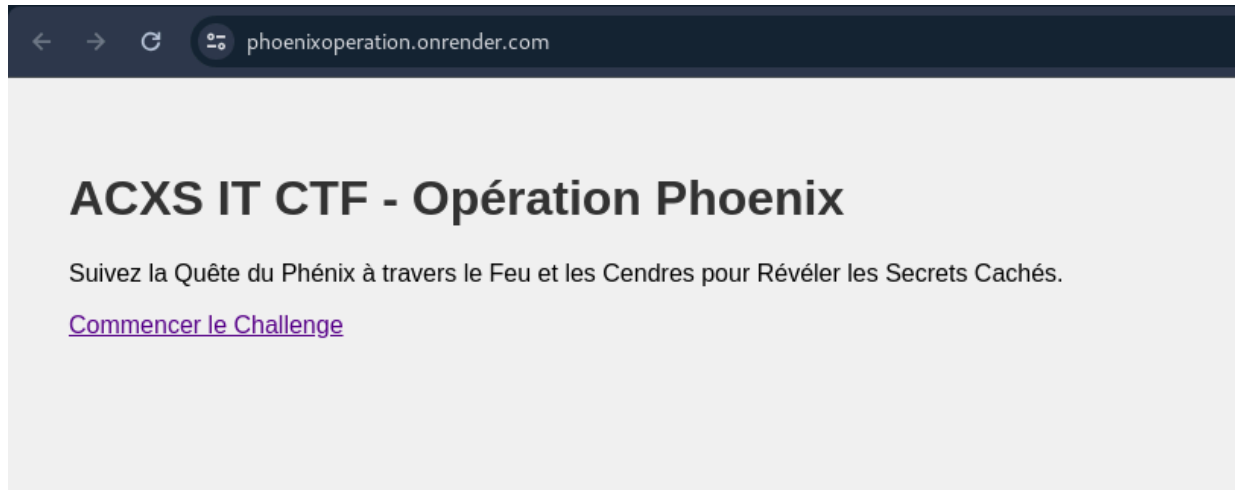


Résolution des solutions de qualification au Phoenix Quest CTF by ACXS IT

Writeup by D0kj4.



On se lance avec la première étape comme on peut le voir dans la capture ci-dessous, il faut réussir à se connecter pour le passer. Pour se connecter, on a besoin des identifiants de connexion dont on ne dispose pas malheureusement. Alors pour ma part j'ai testé une petite injection Sqli en entrant en paramètre:

Nom d'utilisateur: admin' OR '1'='1

Mot de passe: 1234

← → ↻ phoenixoperation.onrender.com/step1

Étape 1 : L'Éveil du Phénix

Nom d'utilisateur :

Mot de passe :

Connexion

Validéeééy

L'étape suivante (étape 2) nous offre deux fichiers dont un contient une clé publique et l'autre deux messages cryptés.

← → ↻ phoenixoperation.onrender.com/step2

Étape 2 : Les Flammes du Savoir

Les flammes du savoir brûlent pour ceux qui osent déchiffrer les énigmes du passé.

Téléchargez les fichiers nécessaires et soumettez le message déchiffré.

- [Télécharger les chiffres](#)
- [Télécharger la clé publique](#)

Message déchiffré :

Vérifier le message

Le contenu des fichiers en question:

```
(kali@dias)-[~/CTF/phoenix]
$ cat public_key.pem
-----BEGIN PUBLIC KEY-----
MCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAJypMXvgPme5WBBG4MPqjjUCAwEAAQ==
-----END PUBLIC KEY-----
```

Ci-dessus la clé publique.

```
(kali@dias)-[~/CTF/phoenix]
$ cat encrypted_messages.b64
aSkjLPQ98/zBYMiyC1Eeg==

Y3pocWVwX2NtX29ubGtzaWRkc18xMDIzNTg3MzY==
```

Ci-dessus les messages chiffrés.

J'ai commencé d'abord à décomposer la clé afin d'identifier le modulus (n) et l'exposant étudié vu la taille de la clé ça devrait par être long.

```
home > kali > CTF > phoenix > decrypt_key.py
1  from cryptography.hazmat.primitives import serialization
2
3  # Chemin du fichier contenant la clé publique
4  file_path = "public_key.pem"
5
6  try:
7      # Lecture de la clé publique depuis le fichier
8      with open(file_path, "rb") as key_file:
9          public_key_data = key_file.read()
10
11     # Charger la clé publique
12     public_key = serialization.load_pem_public_key(public_key_data)
13
14     # Vérification et extraction des composants
15     if hasattr(public_key, 'public_numbers'):
16         numbers = public_key.public_numbers()
17         print(f"Modulus (n) : {numbers.n}")
18         print(f"Exponent (e): {numbers.e}")
19     else:
20         print("Impossible d'extraire les détails de la clé publique.")
21 except FileNotFoundError:
22     print(f"Erreur : le fichier '{file_path}' est introuvable.")
23 except Exception as e:
24     print(f"Une erreur s'est produite : {e}")
25
```

Ci-dessus le code python utilisé pour décomposer la clé.

```
(kali@dias)-[~/CTF/phoenix]
$ python3 decrypt_key.py
Modulus (n) : 208238069164073915652118645967357120053
Exponent (e): 65537
```

Bon voilà on les valeurs de n et e . Passons maintenant au décryptage des messages. J'ai commencé avec le premier message en commençant par le déchiffrer de base64 puis le récupérer sous forme d'entier pour le décoder sur le site <https://www.dcode.fr/chiffre-rsa>. J'ai récupéré la valeur au niveau du "int" pour le décodage.

[kt.gy](#)
[Tools](#)
[Blog](#)
[Twitter](#)

[ASCII](#)
[Hash](#)
[RSA](#)

General

Hash

Misc

ROT

Length: 16 bytes (128 bits)

Reverse

Upload

Download

ij#,ð=ôûÄ`È:p=Dz

ASCII

105 41 35 44 244 61 243 252 193 96 200 178 112 61 68 122

DEC

69 29 23 2c f4 3d f3 fc c1 60 c8 b2 70 3d 44 7a

HEX

151 051 043 054 364 075 363 374 301 140 310 262 160 075 104 172

OCT

aSkjLPQ98/zBYMiyCD1Eeg==

B64

aSkjLPQ98_zBYMiyCD1Eeg

B64URL

NEUSGLHUHXZ7ZQLAZCZHAPKEP|=====

B32

ij%23%2C%C3%B4%3D%C3%B3%C3%BC%C3%81%60%C3%88%C2%B2p%3DDz

URL

ij#,ð=ôûÄ`È:p=Dz

HTML

01101001 00101001 00100011 00101100 11110100 00111101 11110011 11111100 11000001 01100000 11001000 10110010 01110000 00111101 01000100 01111010

BIN

139782537174580612092739074911557731450

INT

Rechercher un outil

RECHERCHE SUR DCode PAR MOTS-CLÉS :
Tapez par exemple 'cesar'

PARCOURIR LA LISTE COMPLÈTE DES OUTILS

Résultats

⚠ Attaque de Wiener: échec

- ✓ P,Q calculé(s) avec N (Décomposition en Facteurs premiers (Auto-Limitée))
- ✓ D calculé(s) avec P,Q,E
- ✓ Déchiffrement avec C,D,N

Affichage limité aux caractères imprimables (autres caractères remplacés par 0)

0NN²uÿr#İ0kBurn

CHIFFRE RSA

Cryptographie > Cryptographie Moderne > Chiffre RSA

DÉCHIFFREMENT DU RSA

Indiquer les nombres connus, laisser le reste vide.

★ VALEUR DU MESSAGE CHIFFRÉ (NOMBRE ENTIER) C= 139782537174580612092739074911557731450

★ CLÉ PUBLIQUE E (GÉNÉRALEMENT E=65537) E= 65537

★ VALEUR DE LA CLÉ PUBLIQUE (NOMBRE ENTIER) N= 20823806916407391565211864596735712053

★ VALEUR DE LA CLÉ PRIVÉE (NOMBRE ENTIER) D=

★ FACTEUR 1 (NOMBRE PREMIER) P=

★ FACTEUR 2 (NOMBRE PREMIER) Q=

★ VALEUR INTERMÉDIAIRE PHI (NOMBRE ENTIER) φ=

★ AFFICHER: ☒ TEXTE CLAIR (CHAÎNE DE CARACTÈRE)
☐ VALEURS CALCULÉES (C,D,E,N,P,Q,...)
☐ TEXTE CLAIR (NOMBRE ENTIER)
☐ TEXTE CLAIR (HEXADÉCIMAL)

► CALCULER/DÉCHIFFRER

LECTEUR DE CERTIFICAT RSA

★ CERTIFICAT (COMMENÇANT PAR -----BEGIN...KEY-----)

► EXTRAIRE LES VALEURS

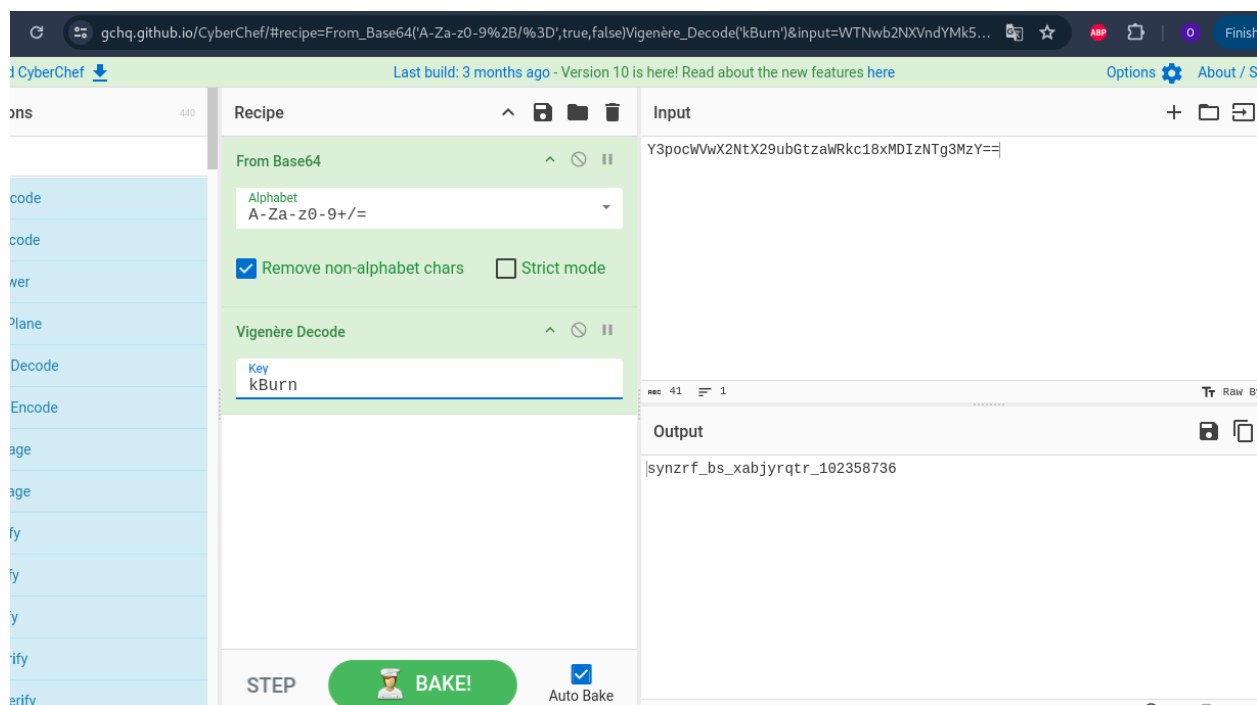
Menu

- ★ Déchiffrement du RSA
- ★ Lecteur de Certificat RSA
- ★ Outils d'aide complémentaires
- ★ Qu'est ce que le chiffre RSA ? (Définition)
- ★ Comment encoder avec RSA ? (Principe de chiffrement)
- ★ Comment décoder le RSA ? (Principe de déchiffrement)
- ★ Comment générer des clés RSA ?
- ★ Comment reconnaître le chiffre RSA ?
- ★ Quelles sont les attaques possibles du RSA ?
- ★ Comment déchiffrer le RSA sans la clé privée ?
- ★ Pourquoi utiliser le nombre e=65537 pour RSA ?
- ★ Comment convertir le nombre en texte clair ?
- ★ Qu'est-ce qu'un certificat RSA ?
- ★ Quand le RSA a-t-il été inventé ?

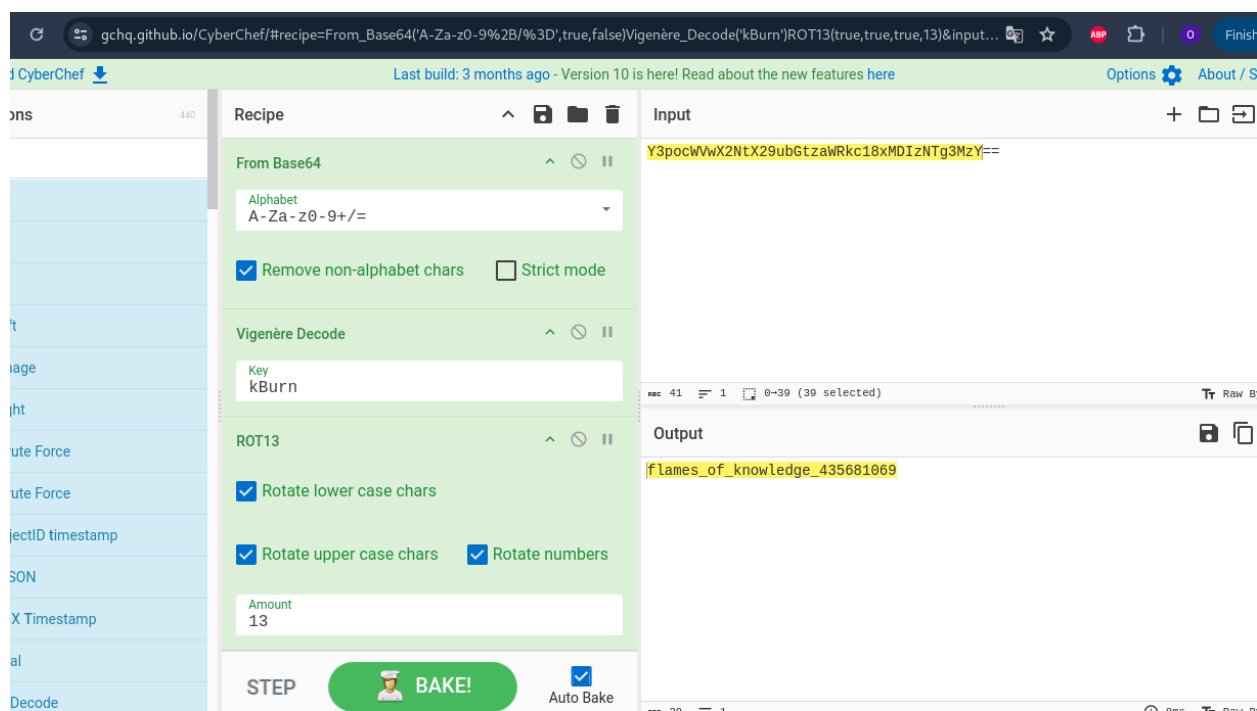
Pages similaires

- ★ Multiplication
- ★ Décomposition en Nombres

Ainsi le premier message donne "kBurn" je doutais toujours du résultats mais pas la suite je me suis retrouvé car le deuxième message n'a pas pu être déchiffré avec la clé. Bon là je me suis dis que kBurn serait probablement la clé pour le déchiffrement et c'est dans cette mentalité que j'ai évolué. Rendez-vous sur <https://gchq.github.io/CyberChef/> pour la suite. J'ai commencé par les algorithmes symétriques et bim Vigenere a sorti un truc chelou...

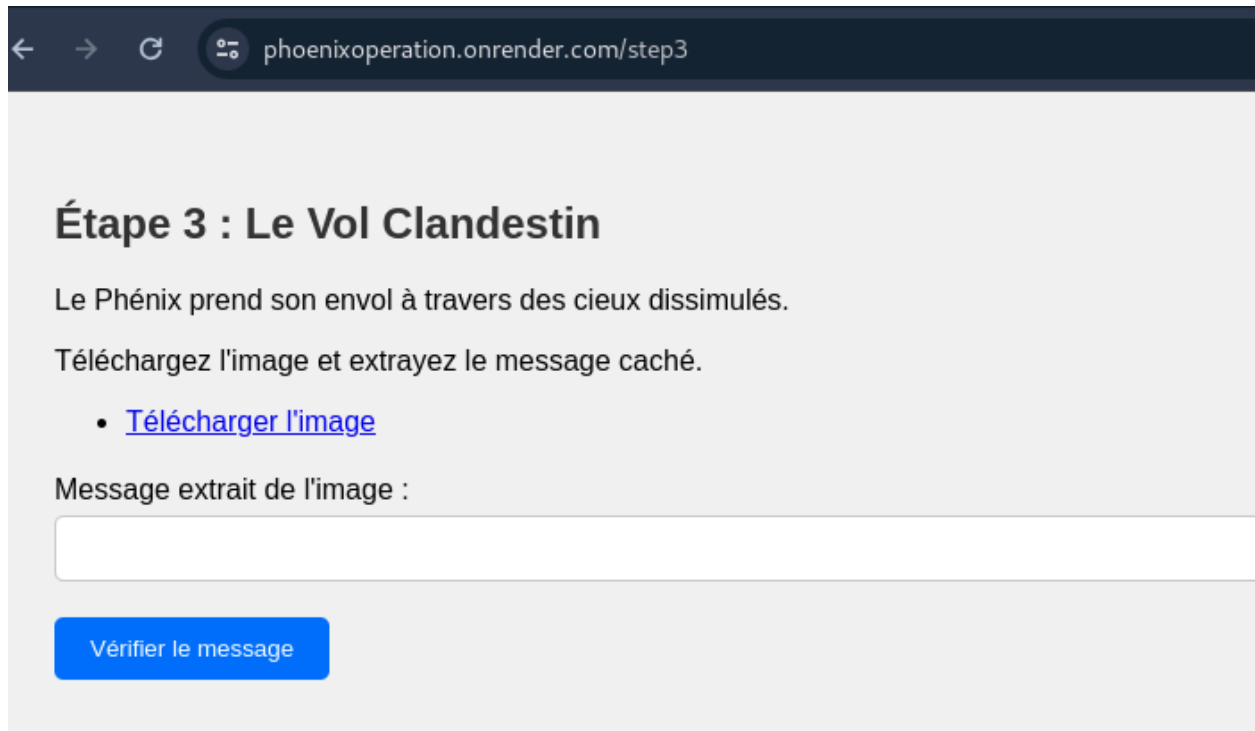


Arrivé à cette étape je me suis lancé sur les algorithmes de décalage et bon je crois c'est ROT13 qui m'a ouvert la porte d'entrée à l'étape 3. J'ai essayé presque toutes les combinaisons pour enfin tomber sur Vigenere → ROT13..



Message chiffré: **flames_of_knowledge_435681069**
Validéééy

Etape 3 now...



The screenshot shows a web browser window with the address bar displaying 'phoenixoperation.onrender.com/step3'. The page content is as follows:

Étape 3 : Le Vol Clandestin

Le Phénix prend son envol à travers des cieux dissimulés.

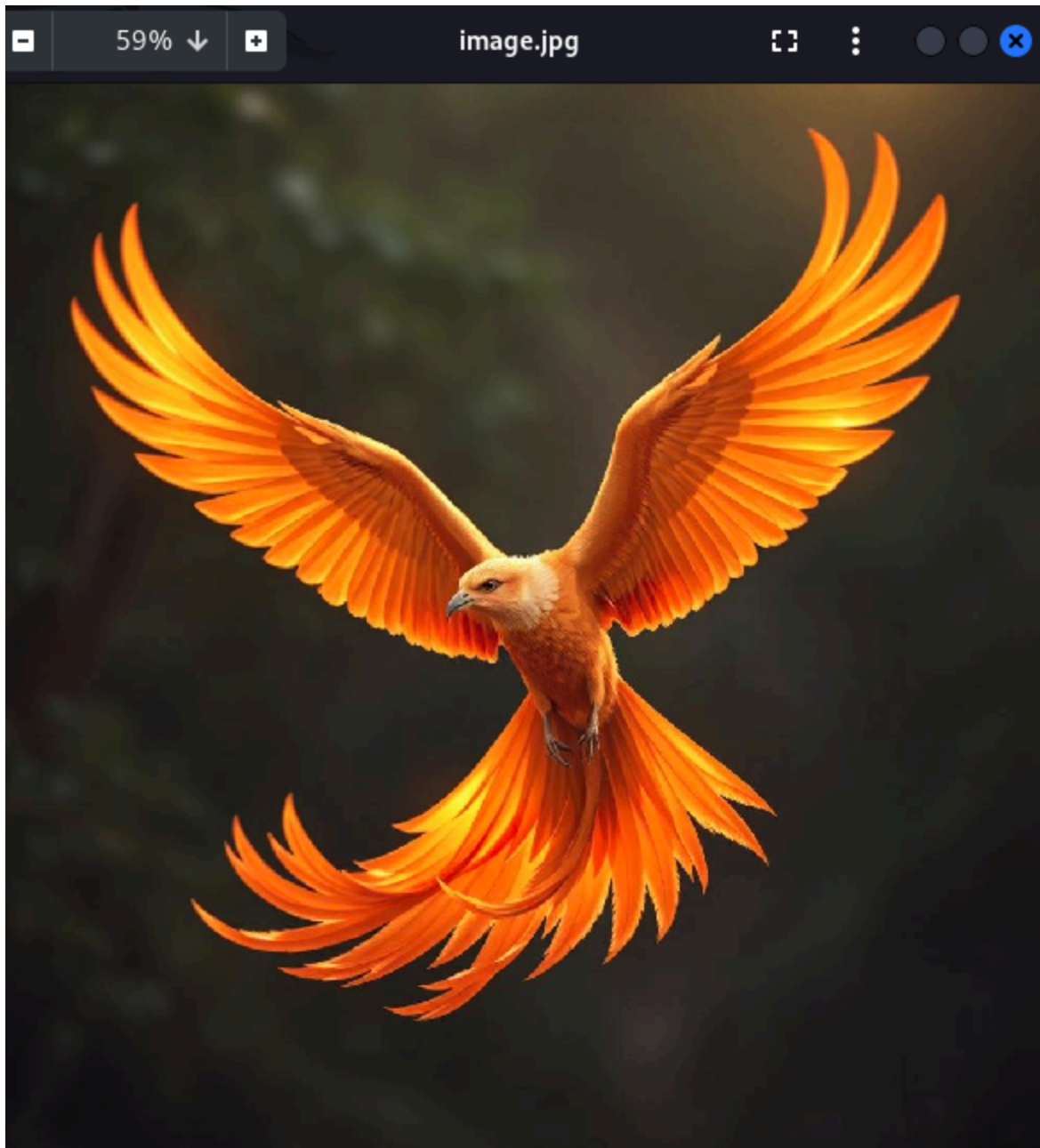
Téléchargez l'image et extrayez le message caché.

- [Télécharger l'image](#)

Message extrait de l'image :

[Vérifier le message](#)

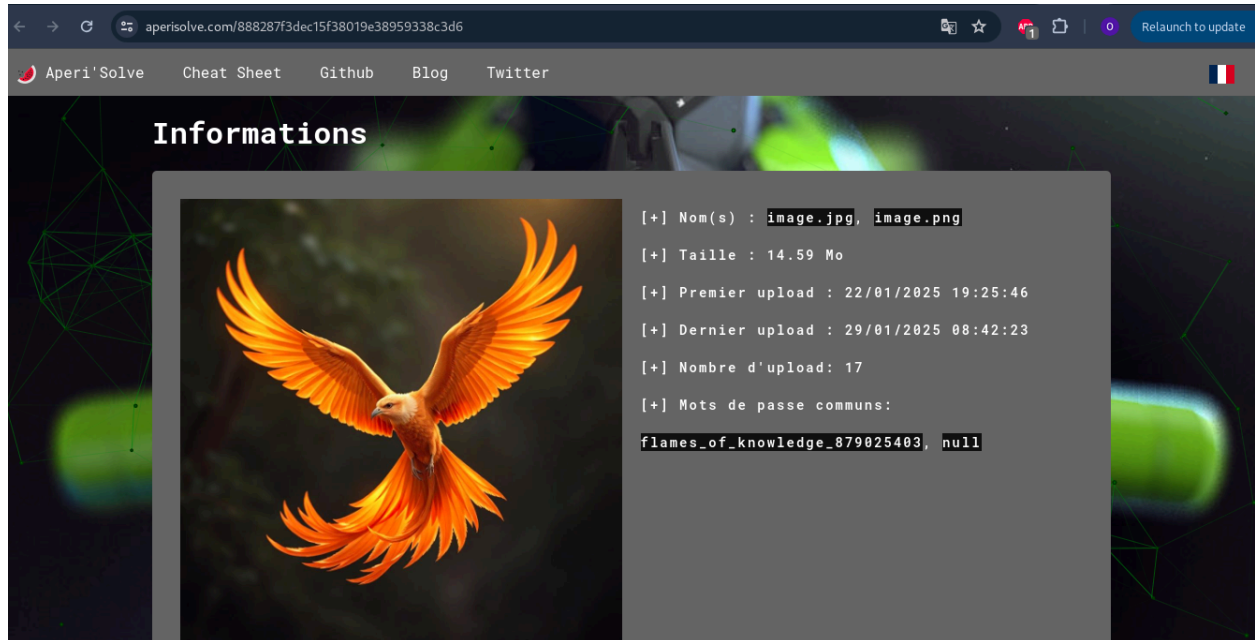
Ici, j'ai à faire à une image PNG (c'est jpg) ils ont mis pour nous embrouillé.



```
(kali@dias)-[~/CTF/phoenix/etape3]
$ ls -lh
total 15M
-rw-rw-r-- 1 kali kali 15M Jan 23 16:37 image.jpg

(kali@dias)-[~/CTF/phoenix/etape3]
$ file image.jpg
image.jpg: PNG image data, 1024 x 1024, 8-bit/color RGBA, non-interlaced
```


On checke **aperisolve** d'abord histoire d'avoir une idée sur le contenu de l'image.



J'ai comme un flag qui n'est pas ce que je cherche mais on le garde pour la suite: **flames_of_knowledge_879025403**

Bref l'image est assez lourde avec **15M** bon on va creuser un peu avec binwalk.

```
(kali@dias)-[~/CTF/phoenix/etape3]
$ binwalk -e image.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 1024 x 1024, 8-bit/color RGBA, non-interlaced
54	0x36	Zlib compressed data, compressed
912674	0xDEd22	RAR archive data, version 5.x
2967146	0x2D466A	RAR archive data, version 5.x
5021618	0x4C9FB2	JPEG image data, JFIF standard 1.01
5094663	0x4DBD07	PNG image, 1024 x 1024, 8-bit/color RGBA, non-interlaced
5094717	0x4DBD3D	Zlib compressed data, compressed
6007337	0x5BAA29	PNG image, 1024 x 1024, 8-bit/color RGBA, non-interlaced
6007391	0x5BAA5F	Zlib compressed data, compressed
6920011	0x69974B	PNG image, 1080 x 1080, 8-bit/color RGBA, non-interlaced
6920065	0x699781	Zlib compressed data, compressed
7409153	0x710E01	Zip archive data, at least v2.0 to extract, name: phoenix.jpg
15299268	0xE972C4	End of Zip archive, footer length: 22

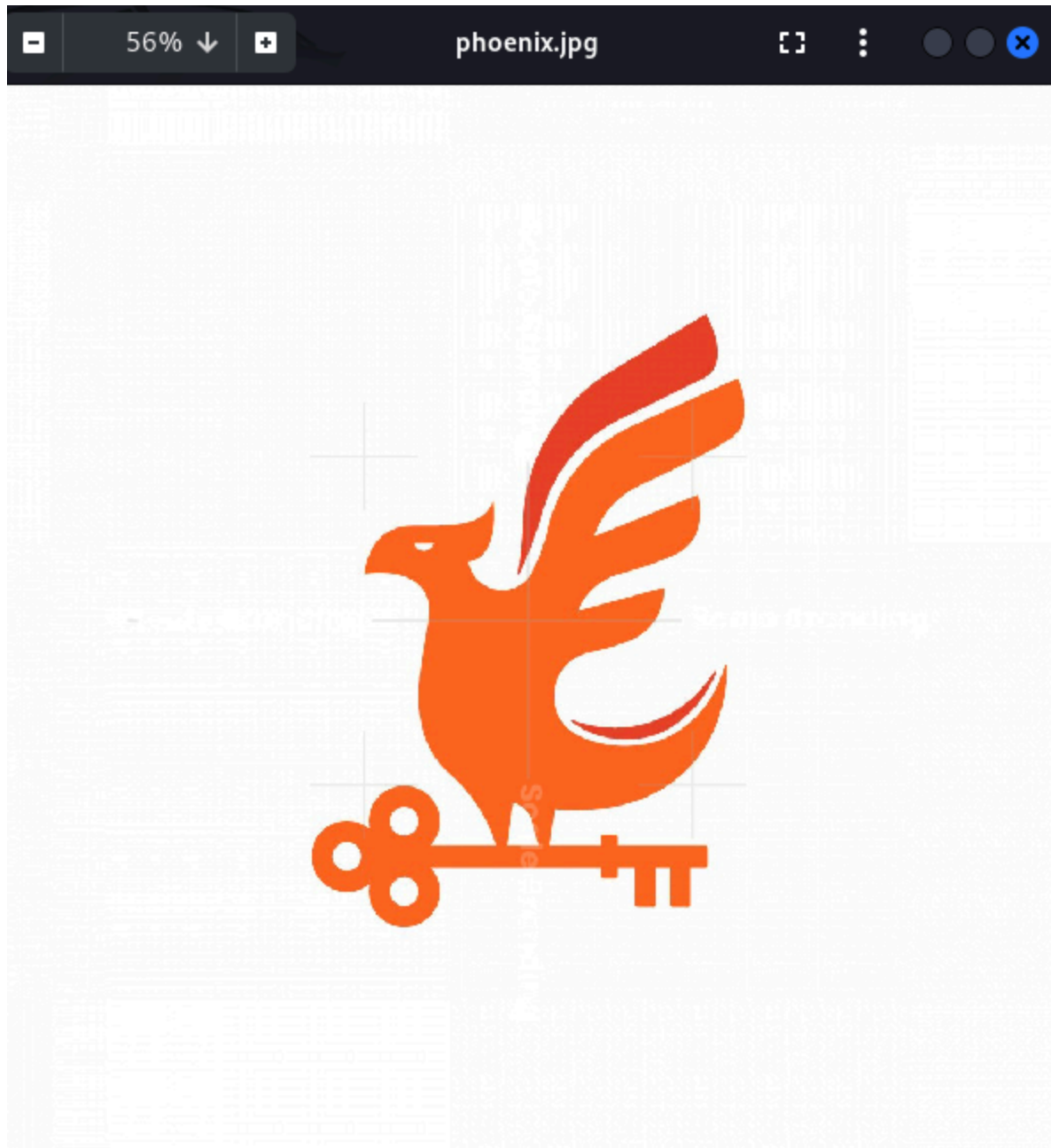
Bon on retrouve une image **phoenix.jpg** et des fichiers RAR contenu dans l'image de base.

```
(kali@dias)-[~/CTF/phoenix/etape3/_image.jpg.extracted]
$ ls -lh
total 82M
-rw-rw-r-- 1 kali kali 12M Jan 29 08:35 2D466A.rar
-rw-rw-r-- 1 kali kali 0 Jan 29 08:35 36
-rw-rw-r-- 1 kali kali 15M Jan 29 08:35 36.zlib
-rw-rw-r-- 1 kali kali 0 Jan 29 08:35 4DBD3D
-rw-rw-r-- 1 kali kali 9.8M Jan 29 08:35 4DBD3D.zlib
-rw-rw-r-- 1 kali kali 0 Jan 29 08:35 5BAA5F
-rw-rw-r-- 1 kali kali 8.9M Jan 29 08:35 5BAA5F.zlib
-rw-rw-r-- 1 kali kali 0 Jan 29 08:35 699781
-rw-rw-r-- 1 kali kali 8.0M Jan 29 08:35 699781.zlib
-rw-rw-r-- 1 kali kali 7.6M Jan 29 08:35 710E01.zip
-rw-rw-r-- 1 kali kali 14M Jan 29 08:35 DED22.rar
-rw-rw-rw- 1 kali kali 7.6M Jan 22 17:30 phoenix.jpg

(kali@dias)-[~/CTF/phoenix/etape3/_image.jpg.extracted]
$ file phoenix.jpg
phoenix.jpg: PNG image data, 1080 x 1080, 8-bit/color RGBA, non-interlaced

(kali@dias)-[~/CTF/phoenix/etape3/_image.jpg.extracted]
$
```

L'image phoenix:



Avant de poursuivre j'ai décompresser les fichiers RAR avec pour mot de passe le faux flag que j'ai trouvé avec aperseolve `flames_of_knowledge_879025403` et j'ai obtenu d'autres images:

```
(kali@dias)-[~/CTF/phoenix/etape3/_image.jpg.extracted]
$ unrar x 2D466A.rar

UNRAR 7.01 beta 1 freeware      Copyright (c) 1993-2024 Alexander Roshal

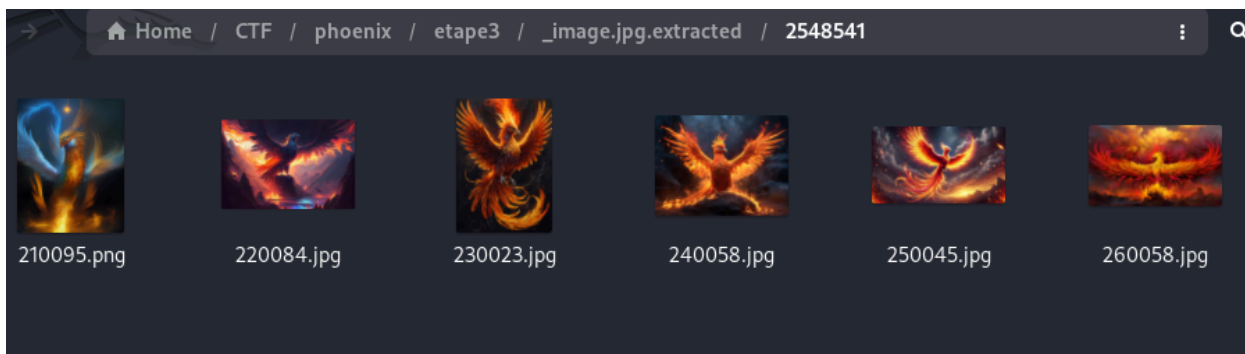
Extracting from 2D466A.rar

Enter password (will not be echoed) for 2548541/210095.png:

Creating      2548541                                     OK
Extracting    2548541/210095.png                           OK
2548541/220084.jpg - use current password? [Y]es, [N]o, [A]ll l

Extracting    2548541/220084.jpg                           OK
2548541/230023.jpg - use current password? [Y]es, [N]o, [A]ll A

Extracting    2548541/230023.jpg                           OK
Extracting    2548541/240058.jpg                           OK
Extracting    2548541/250045.jpg                           OK
Extracting    2548541/260058.jpg                           OK
All OK
```



Bon dans les images retrouvés, il y a des messages chiffrés en base64 comme ceci:

```

(kali@dias)-[~/.../phoenix/etape3/_image.jpg.extracted/2548541]
$ strings 210095.png | tail
72b@1KC
0(J\
Y[BS
&Um!4
)63x
==yXk
+1DDL
&98W
IEND
RkxBR3swNDUyNjU0NTMyMV81ZTE4ODc1ODY1ZTg0ODQ1XzQ1ODUyMjU4fQ==

(kali@dias)-[~/.../phoenix/etape3/_image.jpg.extracted/2548541]
$ strings 220084.jpg | tail
dpz
1)Hu
A0Qaq
1J5F*q1AI
4}>r
OMG1
Zdw(
owHS
*MV\
RkxBR3swNDUyNjU0NTMyMV81ZTE4ODc1ODY1ZTg0ODQ1XzQ1ODUyMjU4fQ==

```

Recipe
^
📁
🗑️

From Base64
^
🔍
⏸️

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars
☐ Strict mode

Input
+
📁
🗑️

RkxBR3swNDUyNjU0NTMyMV81ZTE4ODc1ODY1ZTg0ODQ1XzQ1ODUyMjU4fQ==
RkxBR3swNDUyNjU0NTMyMV81ZTE4ODc1ODY1ZTg0ODQ1XzQ1ODUyMjU4fQ==

Output
📄
🗑️
🔍

|FLAG{04526545321_5e18875865e84845_45852258}|FLAG{04526545321_5e18875865e84845_45852258}|

Finalement le flag était dans l'image:

```
(kali@dias)-[~/.../phoenix/etape3/_image.jpg.extracted/2548541]
$ strings 230023.jpg | tail
x7@ANJ
*T"?:
!u!H
`r9S
%]BB
JD"x
5      )P
G:D#b
b\Qt
RkxBR3toawRkZW5fZmxpZ2h0X2ZvdW5kXzQ1ODAyNTh9
```

Recipe
^
📁
🗑️

From Base64
^
🕒
⏸️

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars
☐ Strict mode

Input
RkxBR3toawRkZW5fZmxpZ2h0X2ZvdW5kXzQ1ODAyNTh9

Output
FLAG{hidden_flight_found_4580258}

Flag: **FLAG{hidden_flight_found_4580258}**

Validéééy

Etape 4 now...

Étape 4 : Les Cendres de la Vérité

Dans les cendres du passé réside la vérité. Téléchargez le fichier à analyser.

- [Télécharger ashes.img](#)

Flag final :

Vérifier le flag

Bon un petit string sur le fichier ashes.img et j'ai un message chiffré en base64

```
(kali@dias)-[~/CTF/phoenix/etape4]
$ strings ashes.img | tail
endobj
5 0 obj
<< /Length 56 >>
stream
RkxBR3t0cnV0aF9pbl9hc2hlc19maW5hbF84MDI1Njg0MTAyNX0NCg==
endstream
endobj
trailer
<< /Root 1 0 R
%%EOF
```

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars ☐ Strict mode

Input

RkxBR3t0cnV0aF9pb19hc2hlc19maw5hbF84MDI1Njg0MTAyNX0NCg==

Output

FLAG{truth_in_ashes_final_80256841025}

Flag: **FLAG{truth_in_ashes_final_80256841025}**
Validééy.

Étape 4 : Les Cendres de la Vérité

Dans les cendres du passé réside la vérité. Téléchargez le fichier à analyser.

- [Télécharger ashes.img](#)

Flag final :

Vérifier le flag

Félicitations ! Vous avez terminé le test avec succès.