CTFs Solution

Chest 1:

File Edit View

Dissect the very essence of the letters within the given legendary fight between Dio and Jotaro! Count them, unravel their lengths, uncover the patterns hidden in plain sight! Use every ounce of your knowledge, every fragment of your intellect but cast aside the meaningless—ignore the special characters and the empty spaces!…but know this—what you face is no modern enigma. No, it is something far older, something timeless… it is CLASSICAL!

Ln 1, Col 1 | 443 characters | 100% | Windows (CRLF) | UTF-8

File Edit View

WRL: Rev zoivzwb gzpvm xziv lu Pzpblrm, gll.

WRL: Klomzivuu nfhg yv srwrmt hlnvdsviv... Szs. Yfg rg wlvhmg nzggvi.

WRL: Blfiv mvcg, Qlgzil!

QLGZIL: Blf yzhgziw. WRL!

WRL: Ls? Blfiv zkkilzxsrmt nv? Rmhgvzw lu ifmmrmt zdzb, blfiv xlnrmt irtsg gl nv? Vevm Gslfts blfi tizmwuzgsvi, Qlhvks, glow blf gsv hvxivg lu Gsv Dliow. orpv zm vczn hgfwvmg hxiznyormt gl urmrhs gsv kilyovnh lm zm vczn fmgro gsv ozhg nlnvmgh yvuliv gsv xsrnv?

QLGZIL: R xzmg yvzg gsv hsrg lfg lu blf drgslfg tvggrmt xolhvi.

WRL: Ls sl! Gsvm xlnv zh xolhv zh blf orpv.

QLGZIL: Liz!

WRL: Gll hold, gll hold! Gsv Dliow rh gsv fogrnzgv Hgzmw. Vevm drgslfg srh kldvi gl hglk grnv, srh hkvvw zmw kldvi uzi vcxvvw gszg lu blfi Hgzi Kozgrmfn.

QLGZIL: Hl rgh gsv hznv gbkv lu Hgzmw zh Hgzi Kozgrmfn. Mlg nfxs izmtv, yfg rnnvmhv kldvi zmw kivxrhv nlevnvmgh.

WRL: R dzmgvw gl gib z orggov gvhg gl urmw lfg qfhg sld nfxs nliv kldviufo Gsv Dliow dzh, xlnkzivw gl blfi Hgzmw.

WRL: Gslfts rg zkkvzih z gvhg dzh sziwob mvxvhhzib.

QLGZIL: Gvhg? Rh gszg dszg blf xzoo qfhg kzggrmt nv, zmw mlg vevm vmlfts gl sfig?

QLGZIL: Dvoo, r tfvhh blf irkkvw nb , bvm kzmgh, gslfts.

WRL: Dsb ziv blf Qlvhgzih hl hgfyylim zylfg zwnrggrmt wvuvzg?

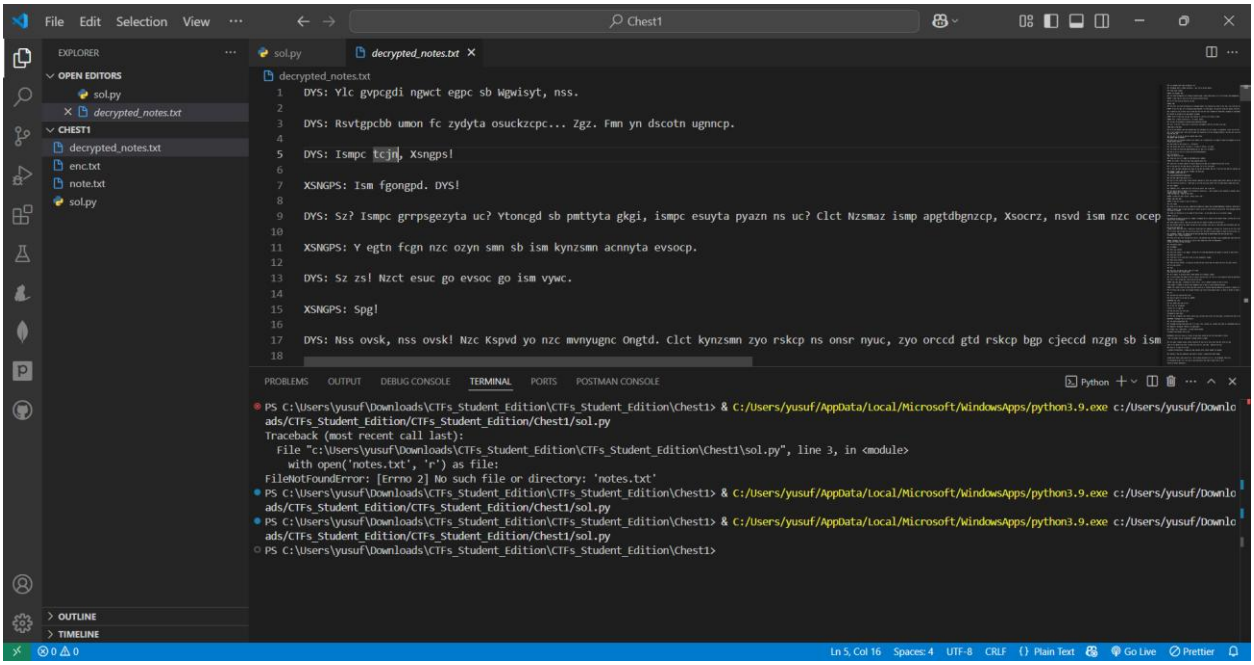
WRL: Snks. R hfkklhv R hszoo irhv gl blfi hsroob kilelxzgrlm, zmw gvhg blf qfhg z yrg nliv.

Ln 1, Col 1 | 19,808 characters | 100% | Windows (CRLF) | UTF-8

Looks like a Caesar cipher. (Dio and Jotaro's script have constant character distances in name dialogues)

K= 19?

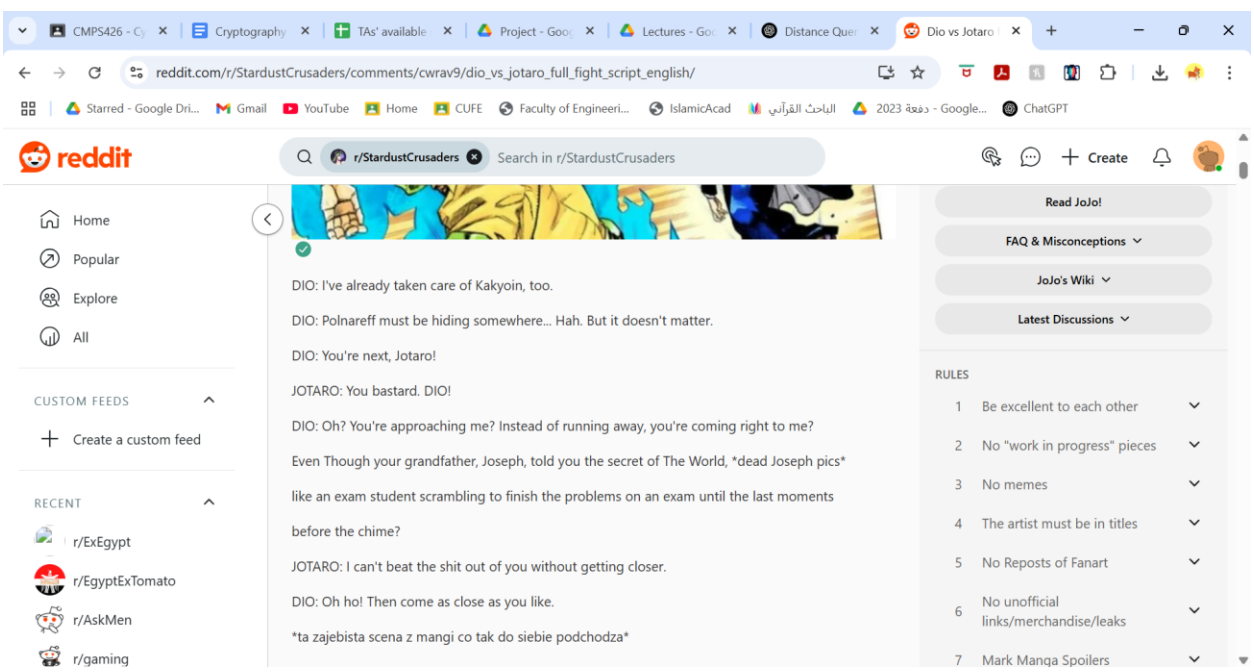
Lets check:



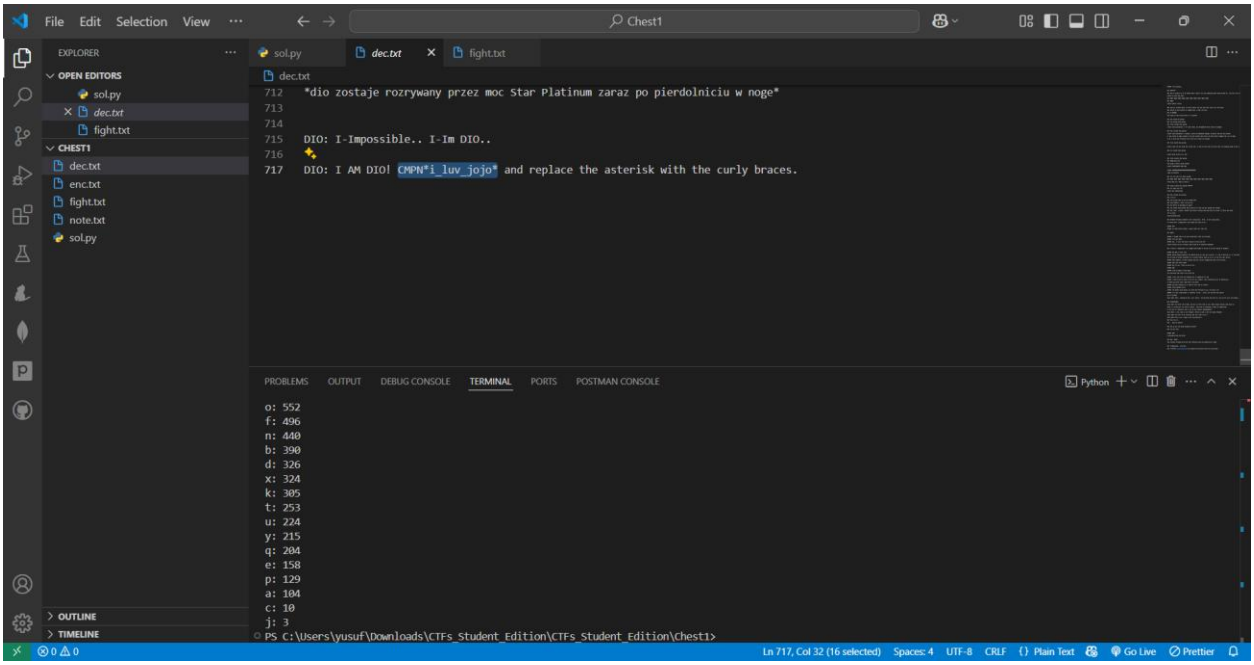
Nope.

HA! A monoalphabetic cipher!

I see ya now



Howa ta2reeban ana gebt l script l asly LOL



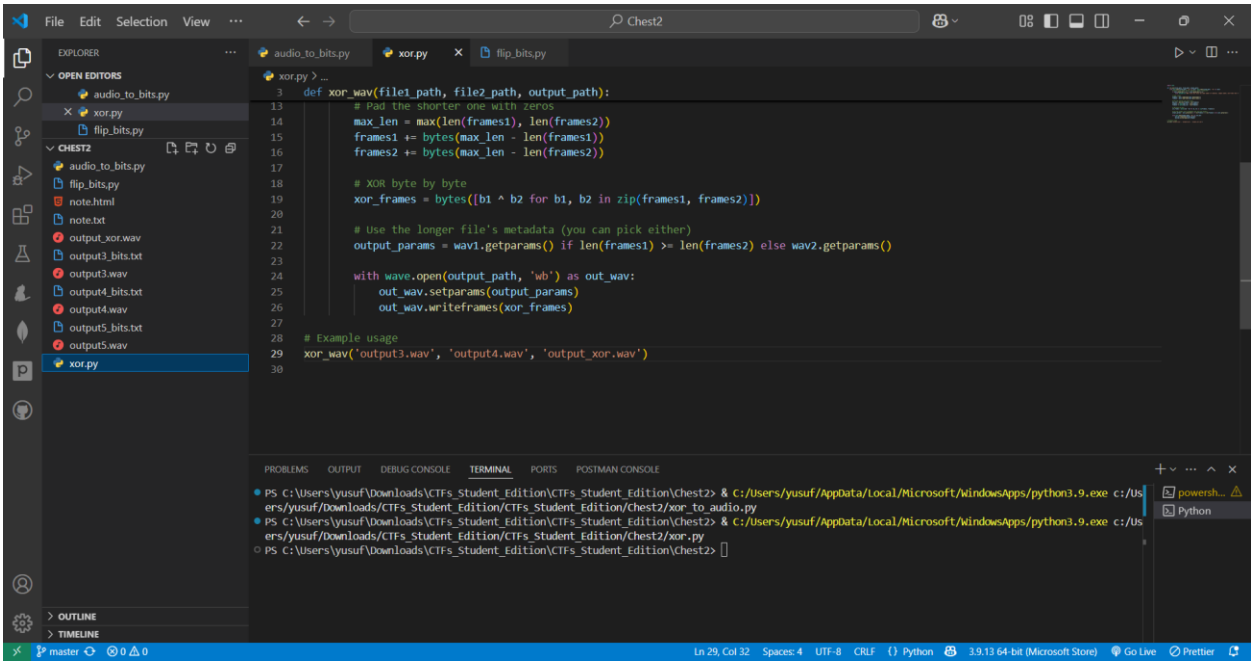
Using python to decrypt the script...

Flag: CMPN{i_luv_jojo}

Chest 2:

Trying to listen to the .wav files didn't help in understanding it.

So lets try xoring them

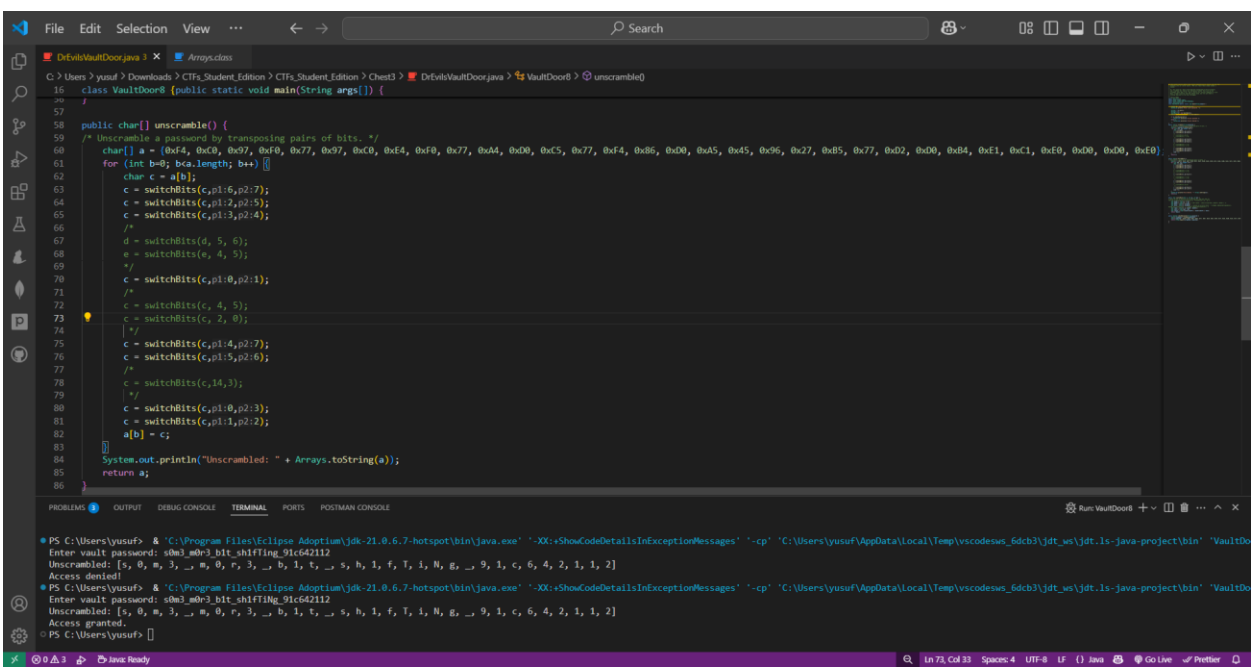


xoring output 3 and 5 gives me a mute file since theyre the same audio

xoring output 3 and 4 however gives me a the key!

Key: CMPN{CYBER_SECURITY}

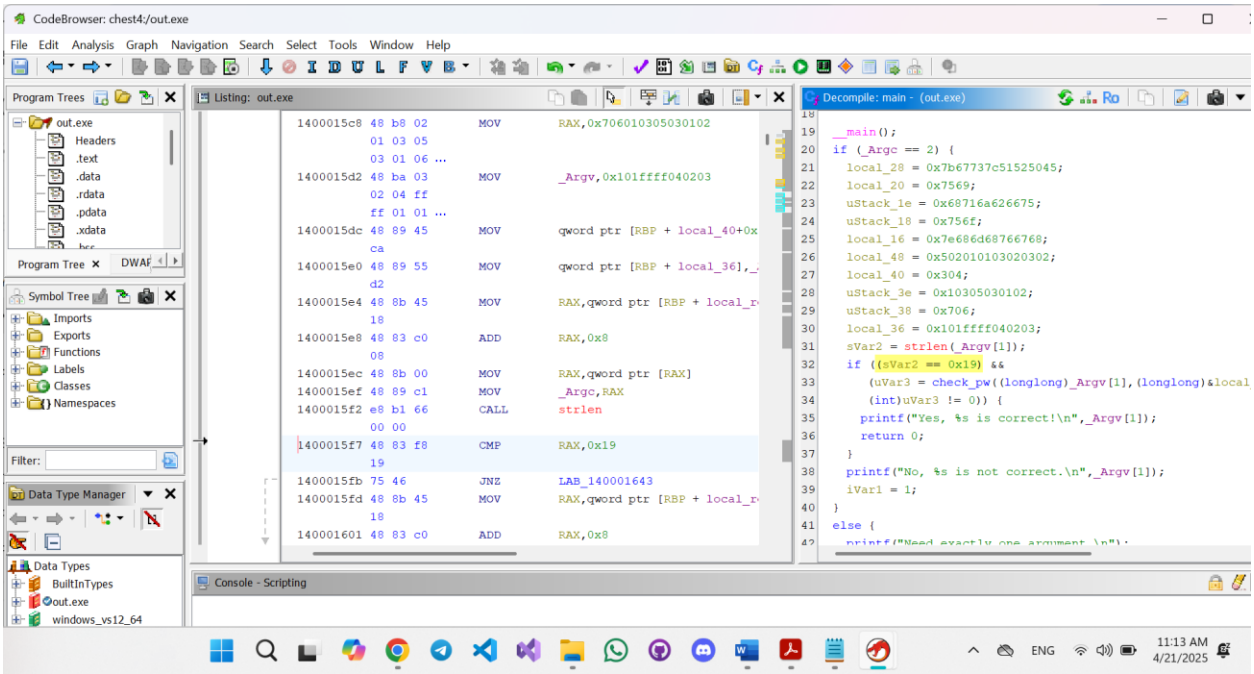
Chest 3:



Created a function to reverse the encryption algorithm and passed the expected passkey

Key: s0m3_m0r3_b1t_sh1fTiNg_91c642112

Chest 4:



Analyzing the decompiled code:
required password length = 0x19=> 25

Check_pw logic:

If input[i] = key[i] – mask[i]

Check_pw params = Argv[1] , &local_28, &local_48

*passing by reference means that the pointer points to the address value of local_28 and local_48

Local_28 = 0x7b67737c51525045 and local_48 = 0x502010103020302

--Since data is stored in memory in little endian

Therefore, param2:

Local_28 = 0x7b67737c51525045

[00] = 0x45

[01] = 0x50

[02] = 0x52

[03] = 0x51

[04] = 0x7C

[05] = 0x73

[06] = 0x67

[07] = 0x7B

what about the rest of the 25 indexes?

-it moves on to the next variables in the memory (local_20->uStack_1e->uStack_18->local_16)

Local_20 = 0x7569

[08] = 0x69

[09] = 0x75

uStack_1e = 0x68716a626675

[10] = 0x75

[11] = 0x66

[12] = 0x62

[13] = 0x6A

[14] = 0x71

[15] = 0x68

uStack_18 = 0x756f

[16] = 0x6F

[17] = 0x75

Local_16 = 0x7e686d68766768

[18] = 0x68

[19] = 0x67

[20] = 0x76

[21] = 0x68

[22] = 0x6D

[23] = 0x68

[24] = 0x7E

[25] = 0x00

Then param3:

local_48 = 0x502010103020302 -> local_40 = 0x304 -> uStack_3e = 0x10305030102 -> uStack_38 = 0x706 -> local_36 = 0x101ffff040203

[00] = 0x02

[01] = 0x03

[02] = 0x02

[03] = 0x03

[04] = 0x01

[05] = 0x01

[06] = 0x02

[07] = 0x05

[08] = 0x04

[09] = 0x03

[10] = 0x02

[11] = 0x01

[12] = 0x03

[13] = 0x05

[14] = 0x03

[15] = 0x01

[16] = 0x06

[17] = 0x07

[18] = 0x03

[19] = 0x02

[20] = 0x04

[21] = 0xFF

[22] = 0xFF

[23] = 0x01

[24] = 0x01

[25] = 0x00

Since input[i] = key[i] – mask[i], while input= Argv[1] , key = &local_28, mask= &local_48

Therefore:
password[i] = key[i] – mask[i]

P[0] = 0x45 - 0x02 = 0x43 => C

P[1] = 0x50 - 0x03 = 0x4D => M

P[2] = 0x52 - 0x02 = 0x50 => P

P[3] = 0x51 - 0x03 = 0x4E => N

P[4] = 0x7C - 0x01 = 0x7B => {

P[5] = 0x73 - 0x01 = 0x72 => r

P[6] = 0x67 - 0x02 = 0x65 => e

P[7] = 0x7B - 0x05 = 0x76 => v

P[8] = 0x69 - 0x04 = 0x65 => e

P[9] = 0x75 - 0x03 = 0x72 => r

P[10] = 0x75 - 0x02 = 0x73 => s

P[11] = 0x66 - 0x01 = 0x65 => e

P[12] = 0x62 - 0x03 = 0x5F => _

P[13] = 0x6A - 0x05 = 0x65 => e

P[14] = 0x71 - 0x03 = 0x6E => n

P[15] = 0x68 - 0x01 = 0x67 => g

P[16] = 0x6F - 0x06 = 0x69 => i

P[17] = 0x75 - 0x07 = 0x6E => n

P[18] = 0x68 - 0x03 = 0x65 => e

P[19] = 0x67 - 0x02 = 0x65 => e

P[20] = 0x76 - 0x04 = 0x72 => r

P[21] = 0x68 - 0xFF = 0x69 => i

P[22] = 0x6D - 0xFF = 0x6E => n

P[23] = 0x68 - 0x01 = 0x67 => g

P[24] = 0x7E - 0x01 = 0x7D => }

P[25] = 0x00 - 0x00 = 0x00 => \0

Key = CMPN{reverse_engineering}