

Write Up Netcomp-CTF



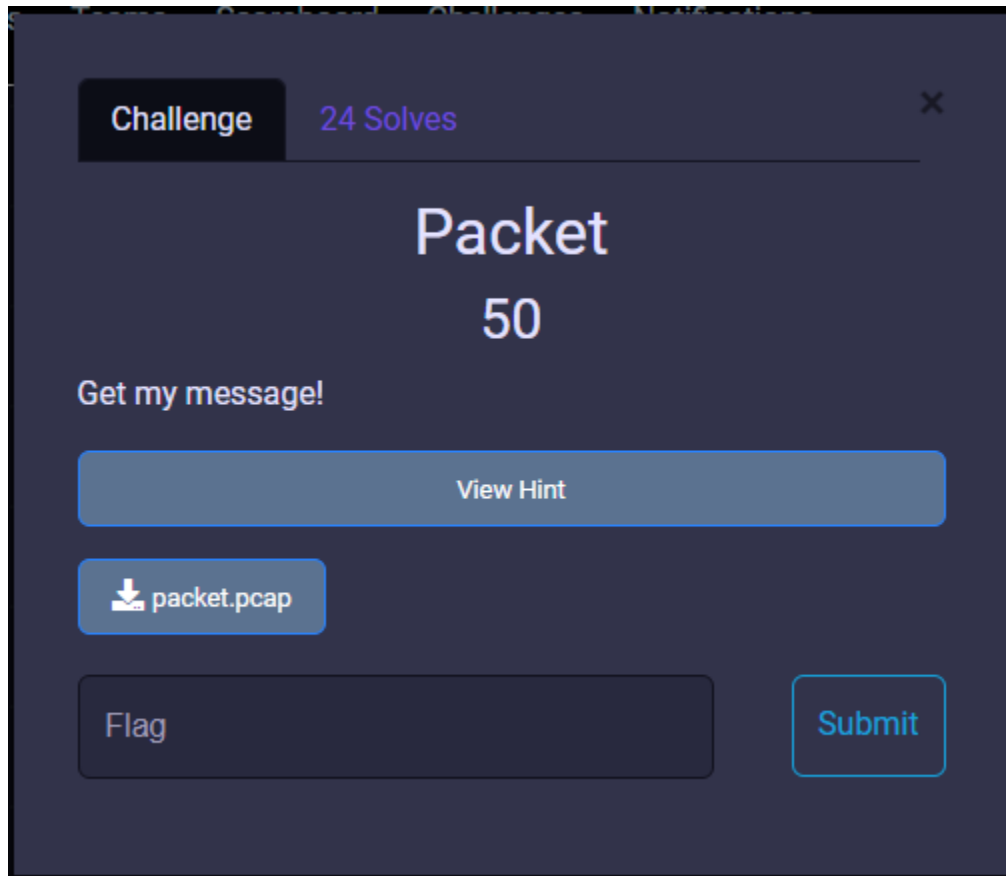
Sembarang Wes

Fikri Muhammad Abdillah
Muhammad Naufal Kurniawan
Kevin Adika Saputra

Forensics	2
Packet (50 pts).....	2
Flag: netcomp{Sending_file_through_icmp}.....	4
bouncies (388 pts).....	5
Flag: netcomp{bouncie_4tt4ck_f0r_r3c0n_op3n_port}.....	6
Learning (490 pts).....	7
Flag: netcomp{H3lp_me_for_buy_wac00o0m_plz_128711FF}.....	11
Cryptography	12
RS 1+1 (50 pts).....	12
Flag: netcomp{easy_RSA_1_plus_1}.....	14
Reverse Engineering	15
ONIC (94 pts).....	15
Flag: netcomp{Buju99_99_jug4_bw4nggg_lo3hhh_h3hh3333}.....	16

Forensics

Packet (24 ~ 50 pts)



Diberikan sebuah file packet.pcap yang berisi capture packet ICMP. Kemudian setelah dianalisa didalamnya terdapat beberapa file png yang bisa ditunjukkan dari icmp data yang terdapat file header 89 50 4e 47 0d 0a 1a 0a.

140 1.234800	192.168.56.111	192.168.56.1	ICMP	100 Echo (ping) reply	id=0x042c, seq=1/256, ttl=64 (request in 140)
148 1.248113	192.168.56.111	192.168.56.1	ICMP	100 Echo (ping) reply	id=0x042e, seq=1/256, ttl=64 (request in 147)
150 1.261809	192.168.56.111	192.168.56.1	ICMP	100 Echo (ping) reply	id=0x0430, seq=1/256, ttl=64 (request in 149)
152 1.310709	192.168.56.111	192.168.56.1	ICMP	100 Echo (ping) reply	id=0x043a, seq=1/256, ttl=64 (request in 151)
154 1.334467	192.168.56.111	192.168.56.1	ICMP	100 Echo (ping) reply	id=0x043f, seq=1/256, ttl=64 (request in 153)
156 1.247020	192.168.56.111	192.168.56.1	ICMP	100 Echo (ping) reply	id=0x0441, seq=1/256, ttl=64 (request in 155)

Frame 4: 100 bytes on wire (800 bits), 100 bytes captured (800 bits)	0000	00 04 00 01 00 06 08 00	27 b8 a0 ac 00 00 08 00
Linux cooked capture v1	0010	45 00 00 54 69 19 00 00	40 01 1f cf c0 a8 38 6f	E...T1...
Internet Protocol Version 4, Src: 192.168.56.111, Dst: 192.168.56.1	0020	c0 a8 38 01 00 00 d5 5f	03 67 00 01 45 5c 98 65	..8....
Internet Control Message Protocol	0030	00 00 00 00 23 23 0f 00	00 00 00 00 89 50 4e 47##...
Type: 0 (Echo (ping) reply)	0040	0d 0a 1a 0a 00 00 00 0d	49 48 44 52 89 50 4e 47
Code: 0	0050	0d 0a 1a 0a 00 00 00 0d	49 48 44 52 89 50 4e 47
Checksum: 0xd55f [correct]	0060	0d 0a 1a 0a	
[Checksum Status: Good]				
Identifier (BE): 871 (0x0367)				
Identifier (LE): 26371 (0x6703)				

Setelah itu dilakukan extract file tersebut dengan kode python sederhana berikut

```
#!/usr/bin/env python3
from scapy.all import *

src_ip = '192.168.56.111'
data = b''

pkts = rdpcap('packet.pcap')
for pkt in pkts:
    if pkt[IP].src == src_ip:
        print(pkt[Raw].load[16:32])
        data += pkt[Raw].load[16:32]

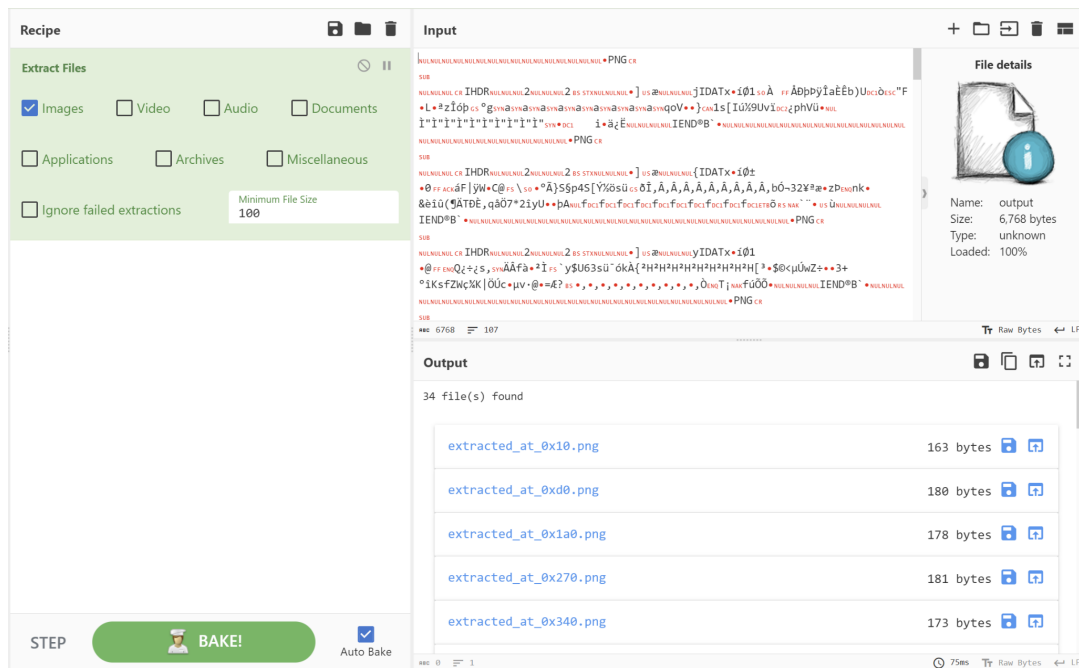
with open("output", 'wb') as f:
    f.write(data)
```

```
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
b'\x89PNG\r\n\x1a\n\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
b'\xe6\x00\x00\x00jIDAT\x9c\xed\xdb\x00\x00\x00\x00\x00\x00\x00'
b'\x0c\x05\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
b'\x1bF\x8bL\x90\xaa\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
b'\x16a\x16a\x16a\x16a\x16a\x16a\x16a\x16a\x16a\x16a\x16a\x16a'
```

Kemudian karena didalam file output terdapat banyak file png, kami menggunakan cyberchef untuk mengextractnya.

```
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000010 89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 |.PNG.....IHDR|
00000020 00 00 00 32 00 00 00 32 08 02 00 00 00 91 5d 1f |... 2 ... 2.....|
00000030 e6 00 00 00 6a 49 44 41 54 78 9c ed d8 31 0e c0 |....jIDATx ... 1..|
00000040 20 0c c5 d0 fe de ff ce 61 c8 ca 62 29 55 11 f2 | .....a..b)U..|
00000050 1b 22 46 8b 4c 90 aa 7a ce f3 fe 1d b0 67 16 61 |."F.L..z....g.a|
00000060 16 61 16 61 16 61 16 61 16 61 16 61 16 61 16 61 |.a.a.a.a.a.a.a.a|
00000070 16 71 6f 56 92 9e 7d 18 31 73 5b 49 fa bd 39 55 |.qoV..}.1s[I..9U|
00000080 76 ef 12 bf 70 68 56 fc 83 00 cc 22 cc 22 cc 22 |v ... phV....".".|
00000090 cc 22 cc 22 cc 22 cc 22 cc 22 cc 22 cc 22 16 98 |.". ". ". ". ". ". |
000000a0 11 09 69 8d e4 bf cb 00 00 00 00 49 45 4e 44 ae |..i.....IEND.|
000000b0 42 60 82 00 00 00 00 00 00 00 00 00 00 00 00 00 |B`.....|
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000d0 89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 |.PNG.....IHDR|
000000e0 00 00 00 32 00 00 00 32 08 02 00 00 00 91 5d 1f |... 2 ... 2.....|
000000f0 e6 00 00 00 7b 49 44 41 54 78 9c ed d8 b1 0a 80 |....{IDATx.....|
```

Writeup CTF - Netcomp

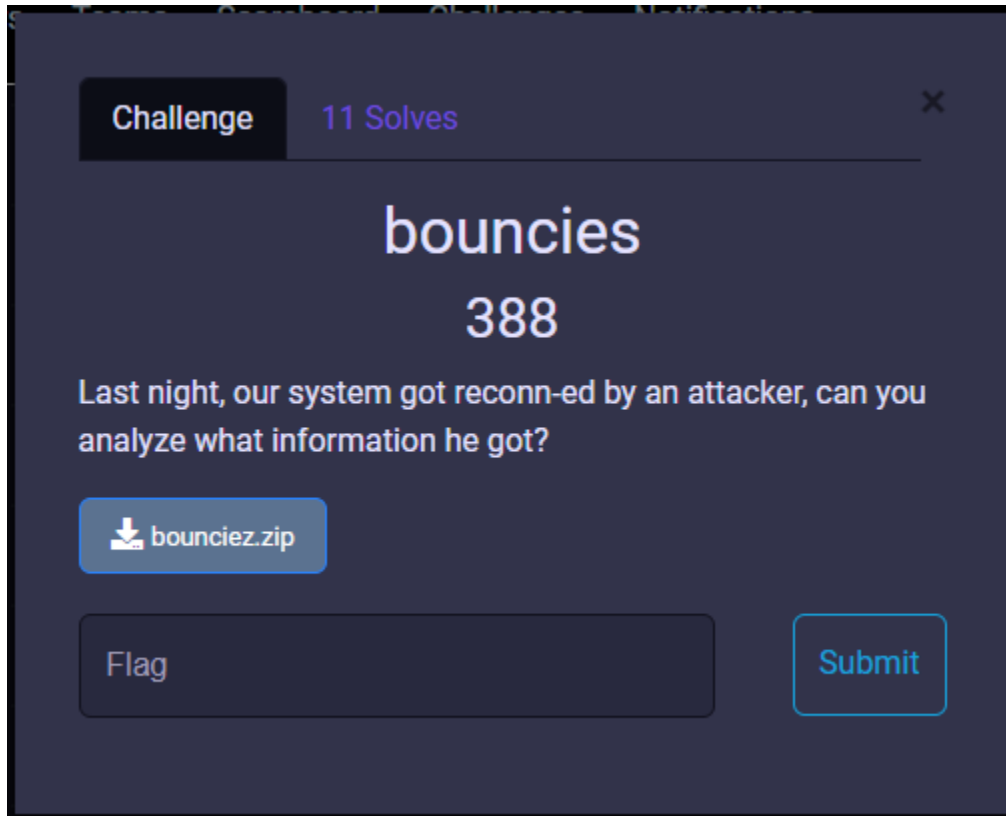


Kemudian tinggal mencatat dan menyusun potongan flag di setiap file png tersebut.



Flag: netcomp{Sending_file_through_icmp}

bouncies (11 ~ 388 pts)



Diberikan sebuah file bounciez.pcapng yang berisi capture packet ketika penyerang melakukan scanning port. Kemudian setelah kami analisa, penyerang menemukan beberapa port yang open, contohnya port 10110, 10101, 10116. Kami coba decode 3 digit terakhir dan ternyata itu merupakan string *net*.

No.	Time	Source	Destination	Protocol	Length	Info
1879	9.980579461	10.10.0.1	10.10.10.2	TCP	74	41292 -> 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1439274265 TSecr=0 WS=128
1880	9.980603367	10.10.10.2	10.10.0.1	TCP	74	21 -> 41292 [SYN, ACK] Seq=0 Ack=1 Win=65168 Len=0 MSS=1460 SACK_PERM TSval=1134184450 TSecr=1439274265 WS=128
1881	9.980620379	10.10.0.1	10.10.10.2	FTP	66	41292 -> 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1439274265 TSecr=1134184450
1882	9.982243516	10.10.10.2	10.10.0.1	FTP	425	Response: 220-..... Welcome to Pure-FTPd [privsep] [TLS]
1883	9.982259046	10.10.0.1	10.10.10.2	TCP	66	41292 -> 21 [ACK] Seq=1 Ack=360 Win=64128 Len=0 TSval=1439274267 TSecr=1134184452
1885	9.982390045	10.10.0.1	10.10.10.2	FTP	81	Request: USER bouncies
1886	9.982403991	10.10.10.2	10.10.0.1	TCP	66	21 -> 41292 [ACK] Seq=360 Ack=16 Win=65280 Len=0 TSval=1134184452 TSecr=1439274267
1889	9.982456561	10.10.10.2	10.10.0.1	FTP	107	Response: 331 User bouncies OK. Password required
1890	9.982485747	10.10.0.1	10.10.10.2	FTP	79	Request: PASS mypass
1891	9.992764195	10.10.10.2	10.10.0.1	FTP	98	Response: 230 OK. Current directory is /
1892	9.992822526	10.10.0.1	10.10.10.2	FTP	90	Request: PORT 10,10,10,2,39,126
1893	9.992915503	10.10.10.2	10.10.0.1	FTP	143	Response: 200-PORT Transfer: From 10.10.0.1 to 10.10.10.2
1894	9.992945881	10.10.0.1	10.10.10.2	FTP	72	Request: LIST
1895	9.993014451	10.10.10.2	10.10.0.1	FTP	96	Response: 150 Connecting to port 10110
1896	9.993041767	10.10.0.1	10.10.10.2	FTP	72	Request: QUIT
1897	9.993150590	10.10.10.2	10.10.0.1	FTP	100	Response: 226-Options: -l
1898	9.993180828	10.10.10.2	10.10.0.1	FTP	133	Response: 221-Goodbye. You uploaded 0 and downloaded 0 bytes.
1899	9.993202035	10.10.0.1	10.10.10.2	TCP	60	41292 -> 21 [RST, ACK] Seq=0 Ack=647 Win=64128 Len=0 TSval=1439274278 TSecr=1134184463

Input

+

📁

🔄

🗑️

📄

110, 101, 116

REC 13 1

Raw Bytes

Output

📄

📄


🔄

📄

net

Kemudian kami extract datanya menggunakan **tshark** seperti berikut untuk mendapatkan semua port yang berhasil ditemukan:


```
tshark -r bounciez.pcapng -Y 'tcp.stream and frame contains "Connecting"' |  
awk '{print $13}'
```

```
Δ kali 2024-01-06 ~   
+ tshark -r bounciez.pcapng -Y 'tcp.stream and frame contains "Connecting"' | awk '{print $13}'  
10110  
10101  
10116  
10099  
10111  
10109  
10112  
10123  
10098  
10111  
10117
```

Setelah itu hasil dimasukan kedalam file **flag.bin** dan decode menggunakan python:

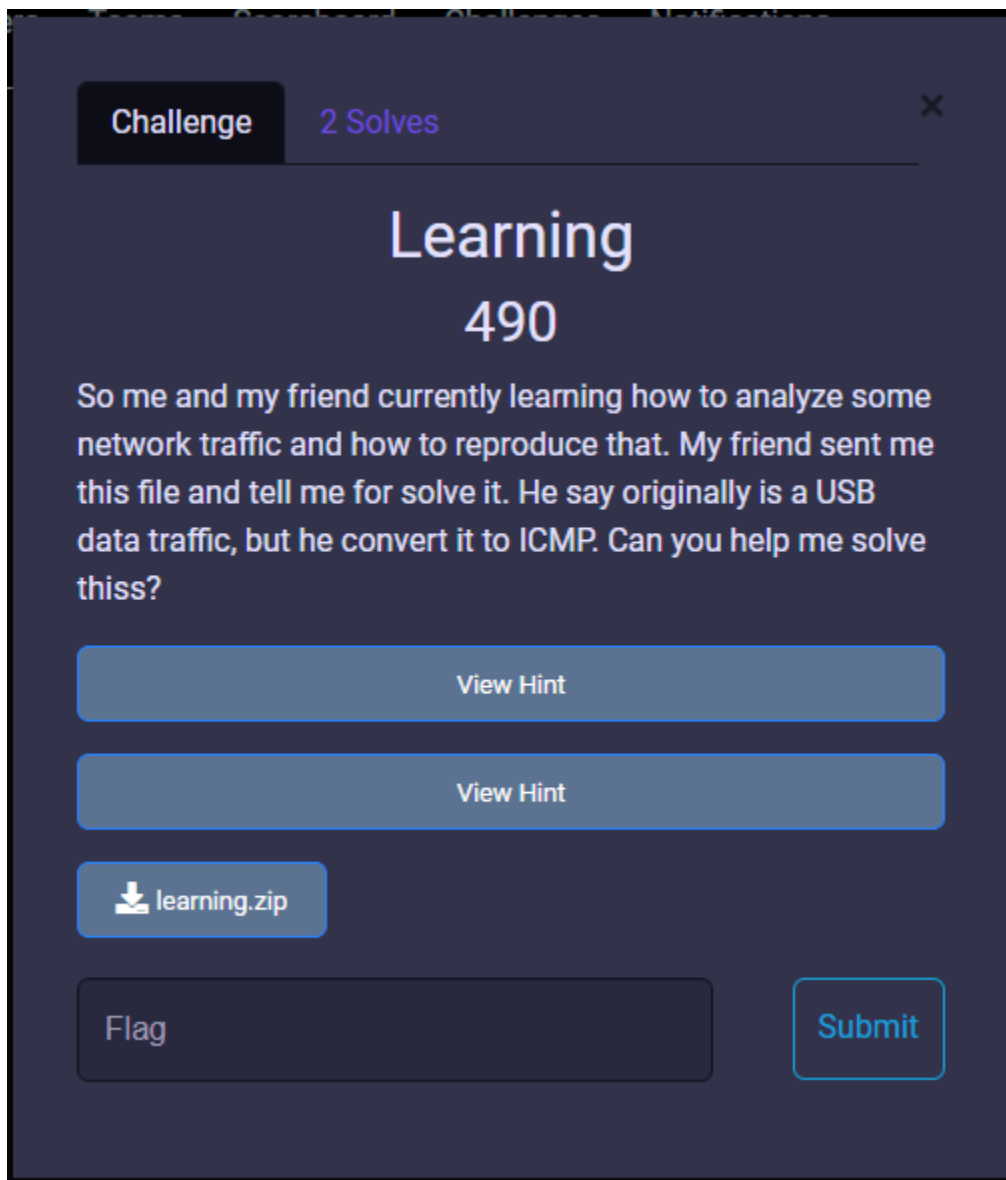
```
with open('flag.bin', 'r') as f:  
    for i in f.readlines():  
        print(chr(int(i[2:])), end="")
```

Hasil:

```
Δ kali 2024-01-06 ~   
→ python3 solve_fix.py  
netcomp{bouncie_4tt4ck_f0r_r3c0n_op3n_port}
```

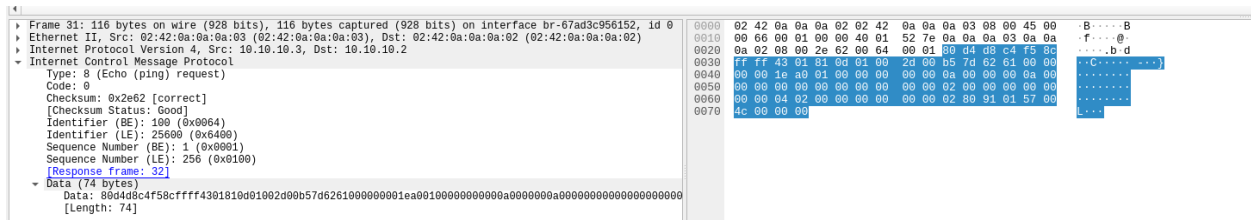
Flag: `netcomp{bouncie_4tt4ck_f0r_r3c0n_op3n_port}`

Learning (2 ~ 490 pts)




Diberikan sebuah file learning.pcapng yang berisi capture packet usb yang telah diubah menjadi packet **ICMP**. Pada hint disebutkan *khususnya trafik USB wacom* setelah mencari beberapa sumber dan kami menemukan sumber berikut <https://blogs.tunelko.com/2017/02/05/bitstcf-tom-and-jerry-50-points/> dan dilihat dari 10 byte terakhir pada beberapa paket **ICMP** memiliki pola yang sama.

Writeup CTF - Netcomp



Pertama kami menggunakan tshark untuk mengekstrak **ICMP** datanya menggunakan tshark.

```
➤ kali 2024-01-06 ~ 
➤ tshark -r learning.pcapng -Y "ip.dst = 10.10.10.3" -T fields -e data.data > capture.data
```

Kami menggunakan filter ip.dst untuk mencegah duplikasi paket, agar yang ditangkap hanya komunikasi 1 arah.

Berikut hasil yang kami dapatkan, data transfer usb adalah 10 byte terakhir:

[illegible]

Kemudian menggunakan script python untuk mengambil nilai x,y,z yang akan di plot nantinya.

```
from pwn import *

with open('capture.data', 'r') as data:
    tmp = data.readlines()
    for i in range(0, len(tmp)):
        x = int(tmp[i][132:136], 16)
        y = int(tmp[i][136:140], 16)
        z = int(tmp[i][140:142], 16)
        if z > 0:
            print(u16(struct.pack(">H", x)), u16(struct.pack(">H", y)))
```

```
kali 2024-01-06 ~  
→ python3 solve.py > /mnt/d/Pentest/draw.txt
```

Kemudian plot menggunakan **gnuplot**

```

└─$ gnuplot
      G N U P L O T
Version 5.4 patchlevel 4      last modified 2022-07-10

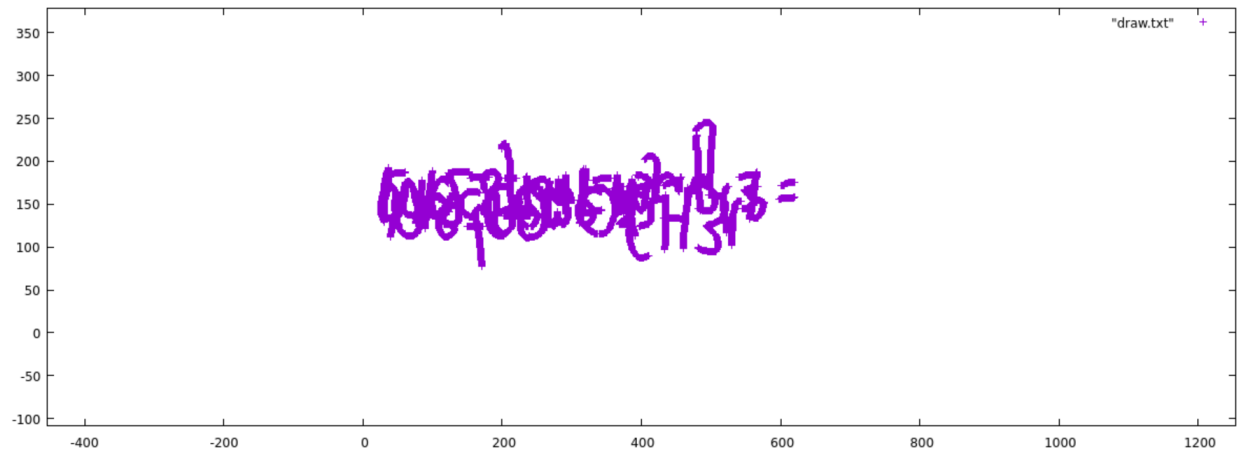
Copyright (C) 1986-1993, 1998, 2004, 2007-2022
Thomas Williams, Colin Kelley and many others

gnuplot home:      http://www.gnuplot.info
faq, bugs, etc:    type "help FAQ"
immediate help:     type "help" (plot window: hit 'h')

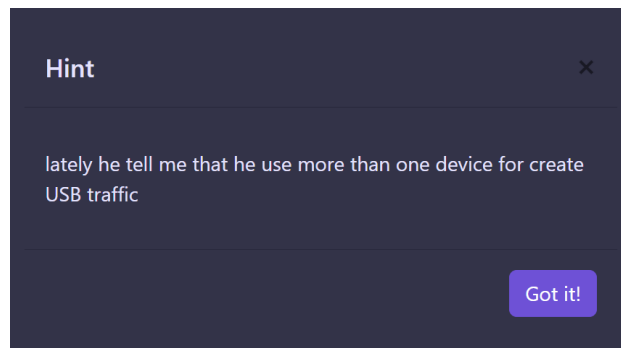
Terminal type is now 'qt'
gnuplot> plot "draw.txt"

```

Dan hasilnya seperti berikut.



Ternyata ada beberapa tulisan yang ditumpuk diatasnya, ini karena sesuai yang dikatakan pada hint yang diberikan yaitu pembuatan USB traffiknya menggunakan lebih dari 1 device.



Kemudian kami mencobanya membagi traffic menjadi 4 bagian dan menyesuaikan offset masing-masing bagiannya.

```
from pwn import *

with open('capture.data', 'r') as data:
    tmp = data.readlines()

    offsets = [
        [
            0, # Start
            (len(tmp)//4)+200], # End
        [
            (len(tmp)//4), # Start
            (len(tmp)//4) + 1200 # End
        ],
        [
```

```

        (len(tmp)//4)+1250, # Start
        (len(tmp)//4)+2600 # End
    ],
    [
        (len(tmp)//4)+2550, # Start
        len(tmp) # End
    ]
]
for i in range(4):
    with open(f'flag_{i}.txt', 'w') as f:
        for i in range(offsets[i][0], offsets[i][1]):
            x = int(tmp[i][132:136],16)
            y = int(tmp[i][136:140],16)
            z = int(tmp[i][140:142],16)
            if z > 0:
                f.write(str(u16(struct.pack(">H",x))) + ' ' +
str(u16(struct.pack(">H",y))))
                f.write('\n')

```

```

kali 2024-01-06 ~
- gnuplot

GNU PLOT
Version 5.4 patchlevel 4    last modified 2022-07-10

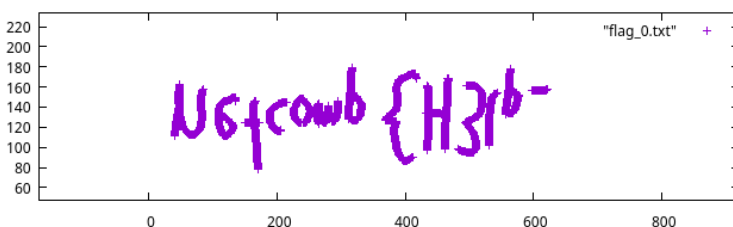
Copyright (C) 1986-1993, 1998, 2004, 2007-2022
Thomas Williams, Colin Kelley and many others

gnuplot home:      http://www.gnuplot.info
faq, bugs, etc:   type "help FAQ"
immediate help:   type "help" (plot window: hit 'h')

Terminal type is now 'qt'
gnuplot> plot "flag_0.txt"
gnuplot> plot "flag_1.txt"
gnuplot> plot "flag_2.txt"
gnuplot> plot "flag_4.txt"
warning: Cannot find or open file "flag_4.txt"
No data in plot

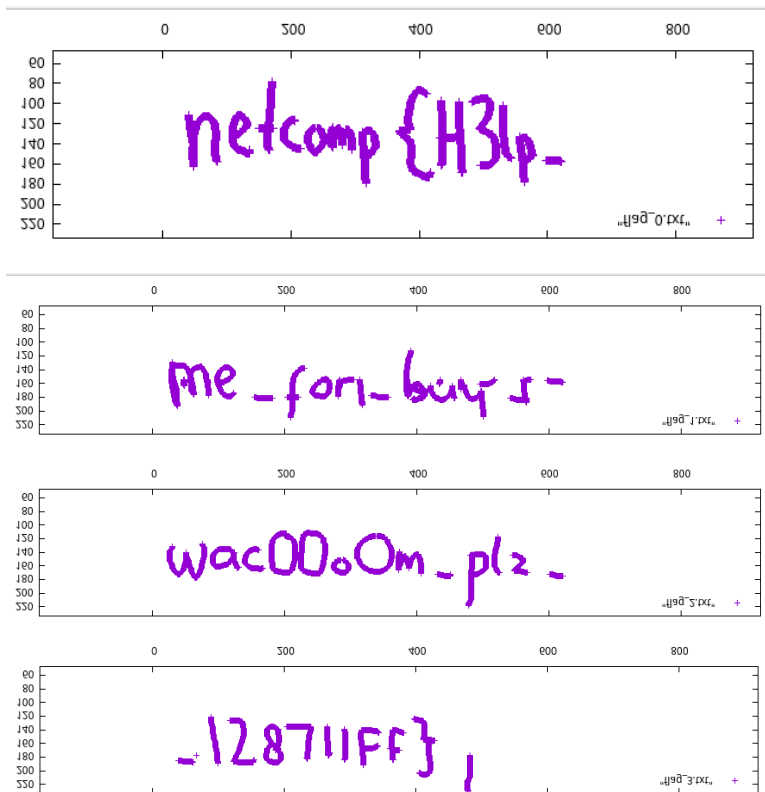
gnuplot> plot "flag_3.txt"
gnuplot>

```



Karena hasilnya gambarnya terbalik, maka harus di flip terlebih dahulu agar memudahkan membaca textnya.

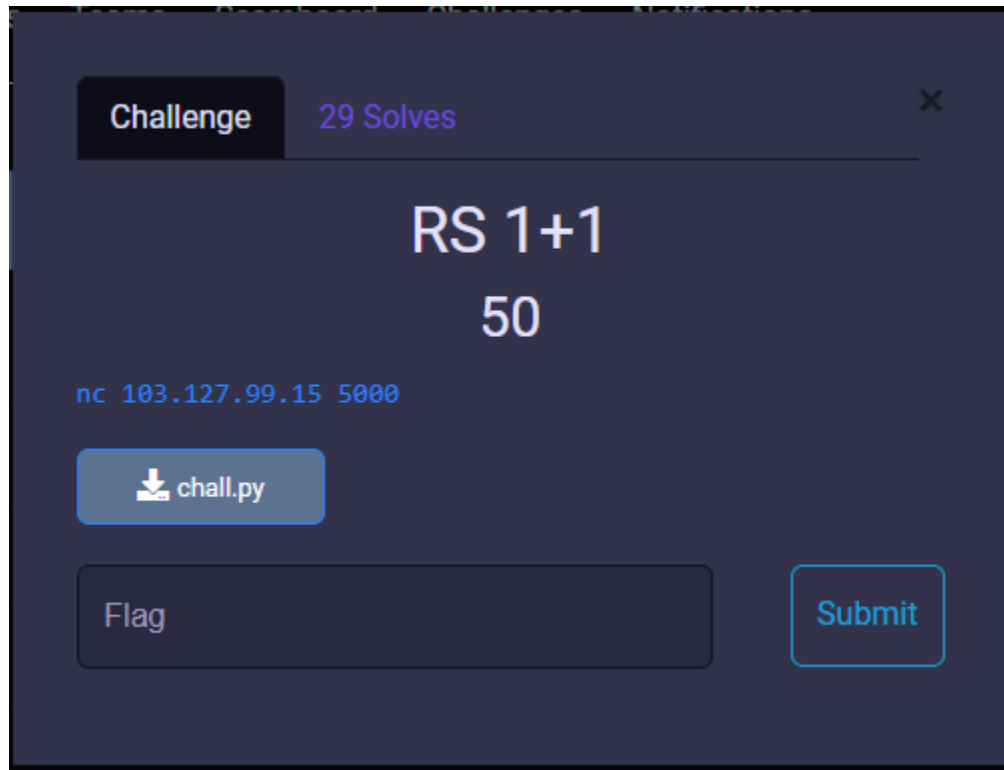
Writeup CTF - Netcomp



Flag: `netcomp{H3lp_me_for_buy_wac00oOm_plz_128711FF}`

Cryptography

RS 1+1 (29 ~ 50 pts)



Kami diberikan koneksi tcp dan file `chall.py`. Langsung saja buka file chall.

Berikut isi dari **chall.py**:

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

from sympy import nextprime
from Crypto.Util.number import *
from random import choice
from flag import flag
import sys

def get_pq(n):
    return getPrime(n), getPrime(n)

def get_token(1):
```

```

        return ''.join(choice('0123456789abcdef') for i in range(1))

correct = 0
while correct < 30:
    token = get_token(32)
    p, q = get_pq(256)
    ppq = p + q
    n = p * q
    e = 0x10001
    m = bytes_to_long(token.encode())
    c = pow(m,e,n)

    print(f'[*] {n = }')
    print(f'[*] {e = }')
    print(f'[*] {c = }')
    print(f'[*] {ppq = }')

    answer = input("[TOKEN]> ")
    if answer == token:
        correct += 1
        print()
    else:
        exit(0)

print(flag)

```

Dapat dilihat dalam file tersebut adalah challenge RSA, dan kita harus memecahkan token yang berupa plaintext dari hasil enkripsi sebanyak 30 kali.

Dari file tersebut kita bisa tahu bahwa kita akan dapat variabel `ppq` yang merupakan hasil penjumlahan dari `p` dan `q`. Karena hal itu kita bisa recovery `p` & `q` dengan menggunakan quadratic equation.

Berikut solver nya, **solve.py**:

```

from decimal import *

from Crypto.Util.number import *
from pwn import *

getcontext().prec = 100000

```

```
HOST = "103.127.99.15"
PORT = 5000

io = remote(HOST, PORT)

def get_pq(n, ppq):
    p = (Decimal(ppq) + (Decimal(ppq) ** 2 - 4 * n).sqrt()) / 2
    p = int(p)
    return p, n // p

for _ in range(30):
    n = int(io.recvline().decode().split(" = ")[1].strip())
    e = int(io.recvline().decode().split(" = ")[1].strip())
    c = int(io.recvline().decode().split(" = ")[1].strip())
    ppq = int(io.recvline().decode().split(" = ")[1].strip())
    print(f"[*] {n = }")
    print(f"[*] {e = }")
    print(f"[*] {c = }")
    print(f"[*] {ppq = }")

    p, q = get_pq(n, ppq)
    print(f"[*] {p = }")
    print(f"[*] {q = }")
    assert p * q == n
    d = inverse(e, int((p - 1) * (q - 1)))

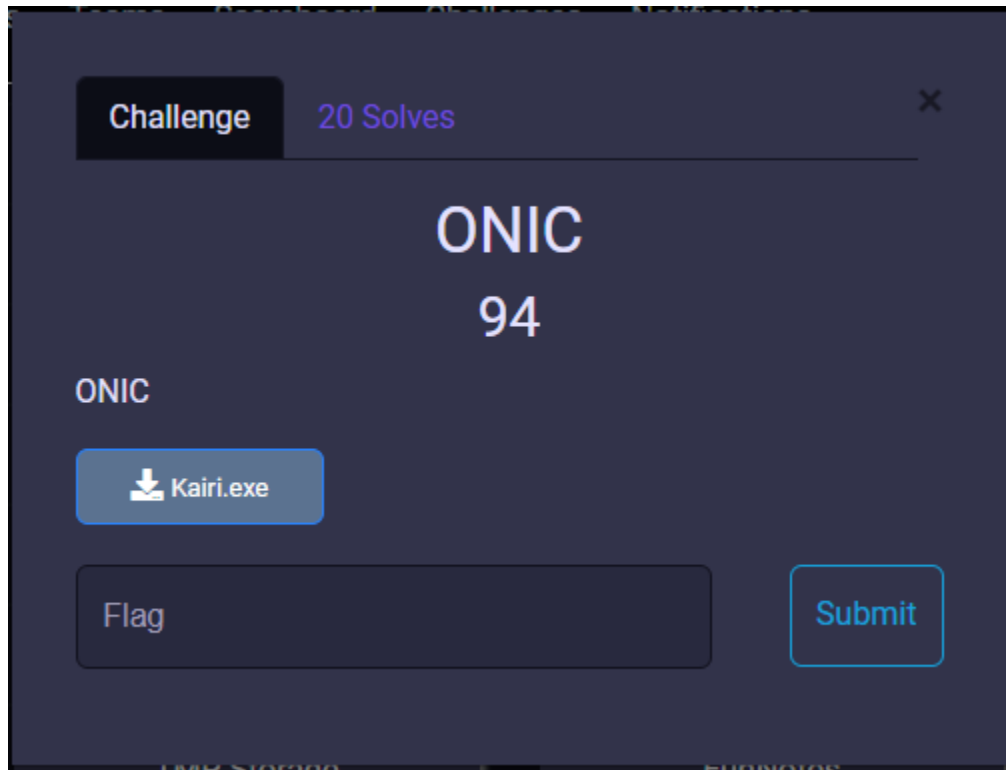
    token = long_to_bytes(pow(c, d, n))
    io.sendline(token)
    io.recvline()

io.interactive()
```

Flag: `netcomp{easy_RSA_1_plus_1}`

Reverse Engineering

ONIC (20 ~ 94 pts)



Diberikan sebuah file Kairi.exe yang merupakan *PE32+ executable*. Kemudian saat coba dieksekusi terjadi error seperti berikut:

```
Error loading Python DLL 'Z:\home\kali\ctf\_internal\python312.dll'.
LoadLibrary: Module not found.
```

Yang berarti File tersebut didalamnya ada sebuah file python, selanjutnya dilakukan extract menggunakan tools pyinstxtractor.py <https://github.com/extremecoders-re/pyinstxtractor>.

```
Δ kali 2024-01-06 ~ [?]
- python3 pyextractor.py Kairi.exe
[+] Processing Kairi.exe
[+] Pyinstaller versions: 2.1+
[+] Python version: 3.12
[+] Length of package: 1286876 bytes
[+] Found 10 files in CArchive
[+] Beginning extraction... please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: flag.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python 3.12 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: Kairi.exe

You can now use a python decompiler on the pyc files within the extracted directory

Δ kali 2024-01-06 ~ [?]
- ls
Kairi.exe  Kairi.exe_extracted  pyextractor.py  README.md  uncompile6.exe
```

Kemudian setelah di extract didalamnya terdapat file flag.pyc, kami coba cetak langsung dan ternyata langsung diberikan flagnya.

Writeup CTF - Netcomp

```

kali@kali:~$ cd Kairi.exe_extracted/
kali@kali:~/Kairi.exe_extracted$ ls
flag.pyc          pyimod01_archive.pyc  pyimod03_ctypes.pyc  pyi_rth_inspect.pyc  PYZ-00.pyz_extracted
pyiboot01_bootstrap.pyc  pyimod02_importers.pyc  pyimod04_pywin32.pyc  PYZ-00.pyz            struct.pyc

kali@kali:~/Kairi.exe_extracted$ cat flag.pyc
♦
♦♦8♦ddlZd♦Zd♦zd♦Zedk(€vy)♦Nc♦l♦t♦|jd♦♦♦||jd♦S)N♦utf-8)♦base64♦   b64encode♦encode♦decode)♦flag♦
                                encoded_flags   ♦flag.py♦
                                                encode_flagr
                                                    S,♦♦♦#♦#♦D♦K♦K♦♦$8♦9♦L♦
                                                                ♦
                                                                    ♦♦♦
                                                                        'e'♦c♦n♦♦

decoded_inputs = r               b64decode)♦
                ♦
                decode_inputs♦♦♦♦$♦$♦]♦3♦:♦:♦7♦C♦M♦
c♦♦♦d}td{ }      t|♦♦||k(R        ♦r
                    td♦ytd♦ydtj
$rt♦♦YyyxYw/Nz/netcomp1BUju99_99_jug4 bw4nggg [o3hh h3hh3333]zMasukkan input yang diencode: z)Bener, Busett Busetttt Kehek aku Bwanggg.z!Salah. aawikwok COba Lagi Br
ohhh.zError: Salah Lurrrrr!!!!.)♦input♦print(binascii.Error)r    ♦
user_inputs = r

```

Flag: `netcomp{Buju99_99_jug4_bw4nggg_Lo3hhh_h3hh3333}`