Most problems here can be simply solved using strings/grep
Some require use of tools such as volatility

Volatility Cheat Sheet

Problem 1: Sad Tall Machine

Ibrahim Sir needs to take 73 makeup classes. Unfortunately, he lost his “Machine” and all he has left is a memory dump captured. In order to find his “Machine” he went to the mighty wizard, sadtallman. The sadtallman requires two pieces of information in order to find the “Machine”. The “MachineType” number and the timestamp for when the dump file was created. Though Ibrahim Sir remembers when he captured the memory the time on his computer was incorrect and sadtallman requires the exact timestamp of the “Machine” during the creation of the dump file.

Flag Format: DUCTF{day_month_year_hour_minute_second_machinetype}

flag: **DUCTF{23_Mar_2012_09:51:53_34404}**

No hint mara kha

Solution:
Simply run this command
python vol.py -f “/path/to/file” windows.info
Timestamp and MachineType is

Problem 2: পরিবেশ পরিবর্তনশীল

আজ সবকিছু হবে বাংলায়

flag: DUCTF{@m1_h01@m_p0t@aK@}

Hint: পতাকা খুঁজুন

Solution 1: Using `strings` and `grep`
strings memdump.mem | grep “POTAKA”

Solution 2: Using `volatility`

The flag is hidden in the environment variables. You can use this command to list all the environment variables
`python vol.py -f memdump.mem windows.envar`
This will list all the environment variables and you can just search for “{“ or “POTAKA” to find the flag.

Problem 3: One More Step

Sadtallman was able to find the “Machine” but the machine had all its files removed by the thief named globalvillageidiot. However, Ibrahim Sir uploaded some of his files to a cloud storage but he forgot the link. Can you recover the link?

Hint 1: There are not many popular cloud storage services for consumers. It’s one of them.

Flag: duCTF{e1d_er_p0r_c1@40654_h03e}

the drive link they need to find is:
“httpsdrive.google.comdrivefolders17c1-5GiGrnOFoR3f0xOjkY65IGcDEHL8usp=sharing”
Adding the proper ‘//’ and stuff:
https://drive.google.com/drive/folders/17c1-5GiGrnOFoR3f0xOjkY65IGcDEHL8?usp=sharing

Solution
This is a two part problem. The first part is finding the google drive link from the memory dump and the 2nd part is actually steganography.
Finding the drive link: This is as simple as it gets. Just search for `strings memdump.mem | grep "drive"` and you should get hundred of results with the incomplete drive link, httpsdrive.google.comdrivefolders17c1-5GiGrnOFoR3f0xOjkY65IGcDEHL8usp=sharing“. It’s very visible that this is a folder link and google drive uses a specific structure for the url. It’s also very easy to guess. After adding the “:, /, ?” you should get the link:
https://drive.google.com/drive/folders/17c1-5GiGrnOFoR3f0xOjkY65IGcDEHL8?usp=sharing

Steganography: Going into this link you’ll find an image file named “final_flag.PNG”. You might think that the flag is simply the model of this car which is the “Toyota C-HR” but you’d be wrong. The flag is hidden in the image file itself.BTW, this is the favorite car of

Hasan Muztoba Sinha. You should send him pictures of this car whenever you can. He'll be happy.

Moving on. If you open the image file using a text editor instead of the image viewer at the end you'll see an encrypted version of the flag appended which is "duCTF{Æ›ÏÄÄÕËÑ£å¾Æ¯®– ¢- ÊÍ -Ñ}". You need to decrypt the flag (excluding the duCTF{..} part of course) but for that you need a **key/passphrase.**
To find the pass/key you'll need to look into the metadata of the image. The simplest way is to use something like `exiftool` and run, `exiftool final_flag.PNG` on this image and you'll see a comment in the exif data, "pass:ajke_class_cancel". You can also use online tools such as this: https://jimpl.com/

After that you have your ciphertext and the key. This is a very Vigenère cipher but with ASCII characters instead of just alphanumeric. Cyclically repeat the key until it matches the length of the ciphertext and subtract the ascii value of each character in the ciphertext by the ascii value of the key and mod by 256.

Here's a python script that does that:

```python
def extended_vigenere_decrypt(encoded_flag, password):
    # Convert the encoded flag and password to their ASCII values
    encoded_ascii = [ord(char) for char in encoded_flag]
    password_ascii = [ord(char) for char in password]

    # Decrypt using extended Vigenère formula
    decrypted_ascii = [
        (encoded_ascii[i] - password_ascii[i % len(password_ascii)]) %
256
        for i in range(len(encoded_ascii))
    ]

    # Convert the decrypted ASCII values back to characters
    decrypted_flag = ''.join(chr(value) for value in decrypted_ascii)
    return decrypted_flag

# Encrypted flag and password
encoded_flag = "Æ›ÏÄÄÕËÑ£å¾Æ¯®– ¢- ÊÍ -Ñ"
password = "ajke_class_cancel"

# Decrypt the flag
decrypted_flag = extended_vigenere_decrypt(encoded_flag, password)
print("Decrypted Flag:", decrypted_flag)
```

Problem 4 (Gorib Village Idiot)

Since the thief globalvillageidiot was unable to keep Ibrahim Sir's "Machine" he went out and bought a Macbook for himself. In the process, he went broke. Now he does not have the money to buy any software and proceeds to download them illegally. However, he ended up downloading a virus. Can you locate where he downloaded the software?
In an astronomically improbable event, the memory dump of globalvillageidiot is exactly the same as the one for Ibrahim sir down to the last BIT.

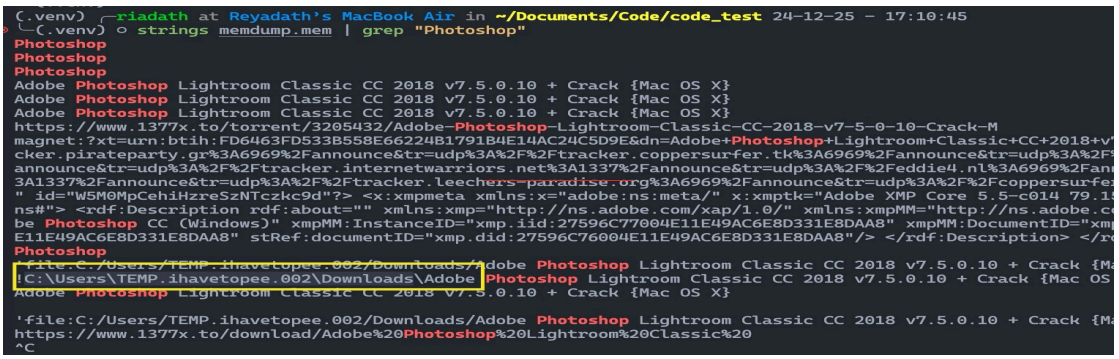
Hint 1: He was using a torrent service to download said software

flag: **DUCTF{C:\Users\TEMP.ihavetopee.002\Downloads}**

Solution 1

Firstly, you can simple search the dump file using **`strings memdump.mem | grep "crack"`**. The a bit of scraping through the output should reveal that the pirated software is "Adobe Photoshop Lightroom Classic CC". The actual location is already present in the current output but the large amount of strings might be hard to navigate.

To narrow down your search you can simply search for, `strings memdump.mem | grep "Photoshop"` and within the first few lines.



Solution 2

A more systematic approach would be to go through the process list using the command `**vol.py -f "/path/to/file" windows.pslist**` and see that the user is running "qbittorrent".
From there you can get the **PID** of the process and get the memory dump of that specific process i.e. qbittorrent. From there you can run the command, `**vol.py -f "/path/to/file" -o "/path/to/dir" windows.memmap --dump --pid <PID>**` to get a more concise dump of and from there it is very easy to deduce the program (Photoshop) that the user is trying to download and also the download location.

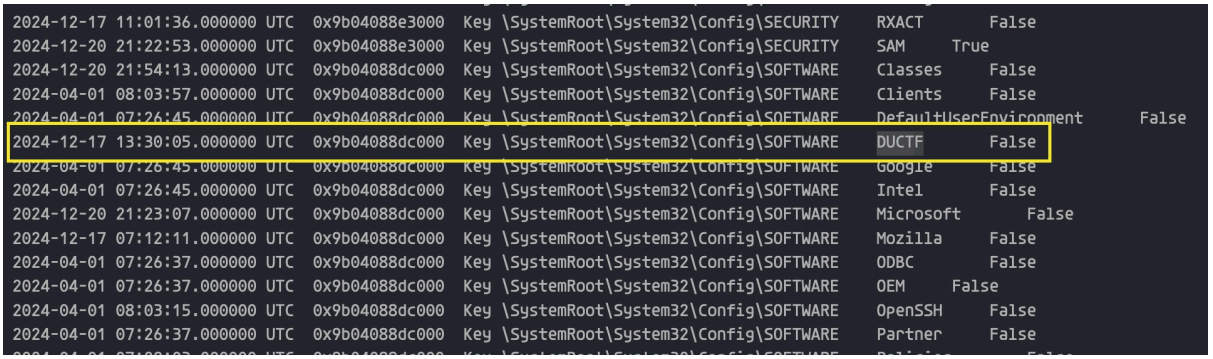
Problem 5 (Gorib Village Idiot 2)

globalvillageidiot has turned to a word of crime. He now robs super shops and steals the all the cash from the "registry". sadtallman has to catch this notorious thief. Can you find the "Address" and the "Time" of the robbery?

Flag Format: duCTF{YYYY-MM-DD_HH:MM:SS_TIMZONE_Address}
Flag: **duCTF{2024-12-17_13:30:05_UTC_0x9b04088dc000}**

Solution

It should be very very obvious from the flag format and the statement that the flag is hidden in the **registry files of windows.**
Just simply run the command,
`**python vol.py -f "/path/to/file" windows.registry.printkey**`
From there just search "DUCTF" and it's all there.



Problem 6 (F**k Ubuntu)

https://youtu.be/tuhzVDc0Slg?t=784

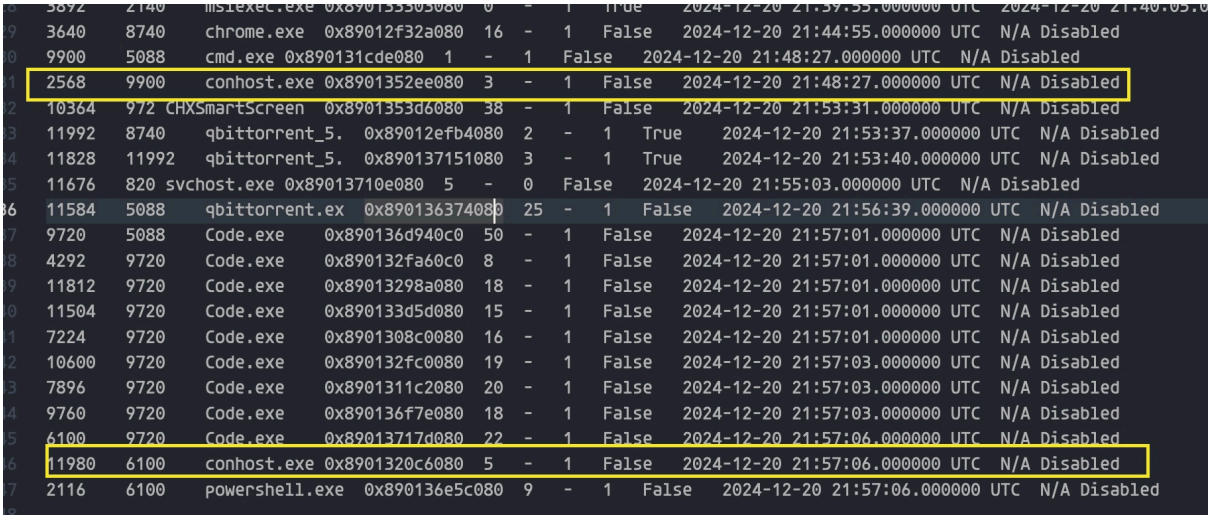
Flag Format: duCTF{PID_FileName_Offset_YYYY-MM-DD_HH:MM:SS}

flag 1: **duCTF{2568_conhost.exe_0x8901352ee080_2024-12-20_21:48:27}**
flag 2: **duCTF{11980_conhost.exe_0x8901320c6080_2024-12-20_21:57:06}**

Solution

The entire clip provided is about the windows “Console Host” or “conhost”. A simple look through the process list should reveal all the necessary info (i.e. PID, Filename etc.)
Use command: `**vol.py -f “/path/to/file” windows.pslist**`

BOTH ANSWERS WILL BE ACCEPTED AS THE FLAG IN THIS PROBLEM



Osint

Problem 1 (Delay, Deny, Depose)

https://vk1cd314.github.io/
Format: duCTF{num1_num2}
[Rounded to 3 decimal places]
flag: duCTF{40.762_-73.979}

coordinates up to three digits more or less the same everywhere
exact coordinates: 40.762259204902534, -73.97919010418326

Problem 2 (The Bag of Death)

duCTF{url}

flag:
duCTF{https://www.peakdesign.com/global/products/everyday-backpack}