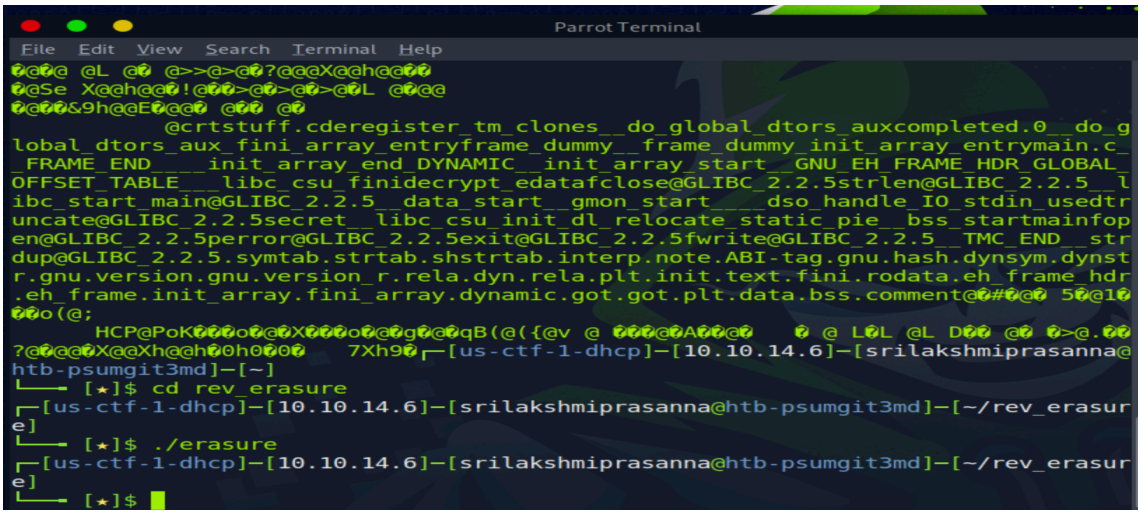
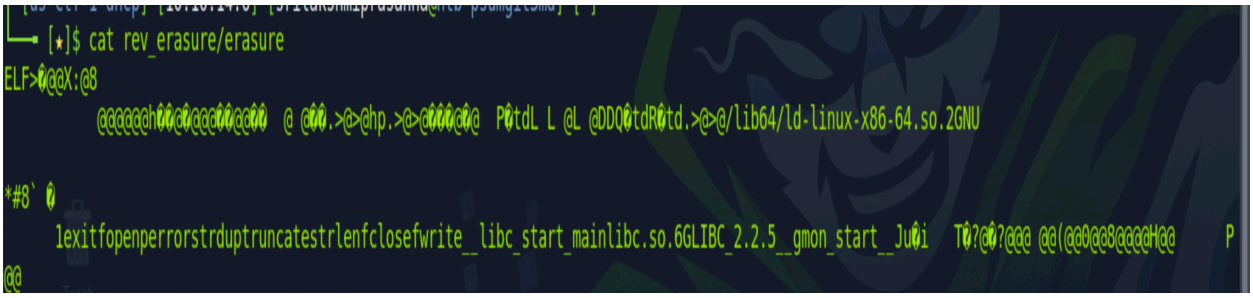
Erasure Challenge

Step1: Downloaded the zip file and unzipped then checked the contents of the file and found it is ELF file. Then tried to execute the erasure file normally as ELF file which gave nothing as shown below.

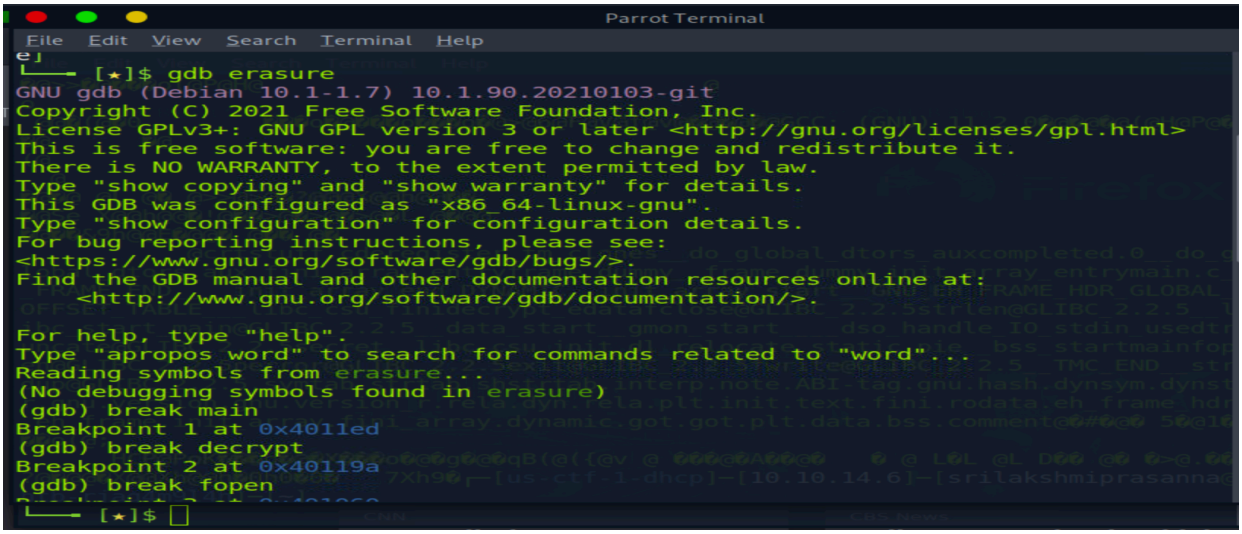


Step2: From description understood, I understood that there might be a function that could be deleting/erasing the output. So I inspected the ELF file content and found words like exit, fopen, perror, fclose, fwrite, main and decrypt and wflag.txt as shown in the below screenshots.

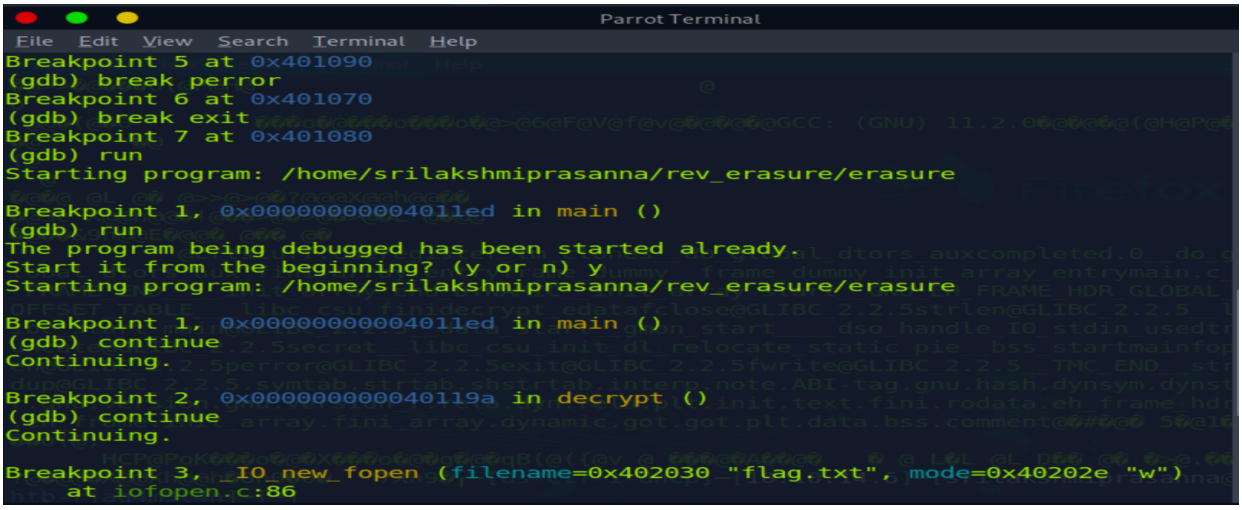


Step3: Then thought they could be functions names in the whole code. So I used GDB(GNB Debugger) which could break the code files at certain points

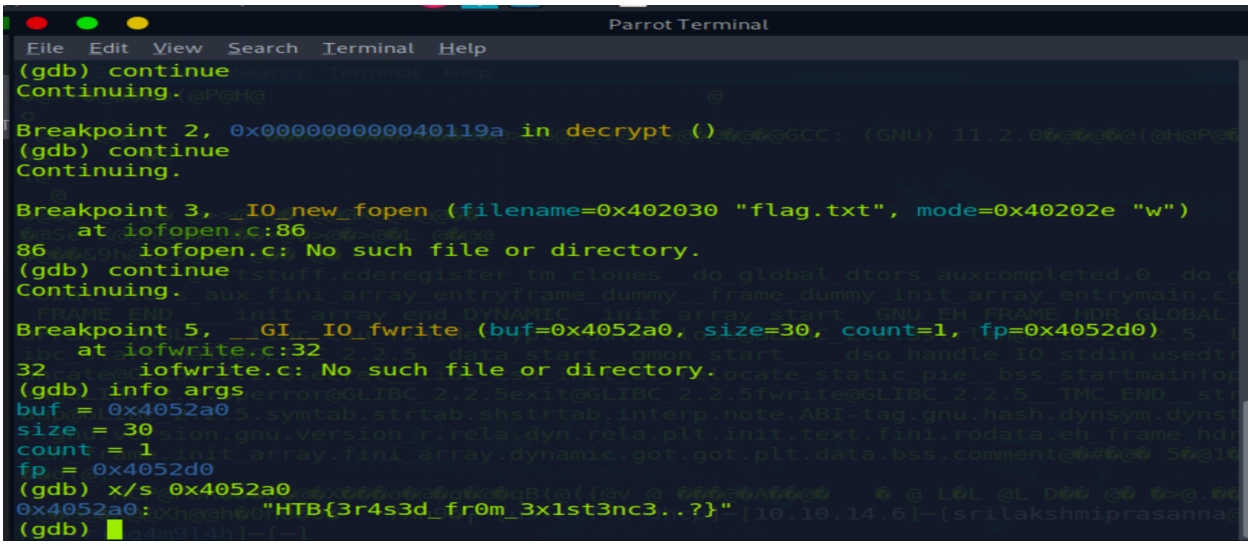
such as line by line or function by function and then executes. Hence, I broke the code using the functions names found above.



Step4: Then started running code from one function to another functions and found flag.txt as shown in the below screenshot.



Step5: I still continued the execution and got the contents of the flag.txt file with help of ‘fwrite’ function which gave buffer(contains data to be written by ‘fwrite’ function) and accessed buffer and got flag as shown in the below screenshot.



Risk mitigation strategy:

This program might be created with code to erase the information to stop others seeing the sensitive data. But with the help of GDB, I was able to disclose the sensitive data.

We can mitigate this problem by providing least privileges for file access. If the file don't have 'read' and 'execute' permission for others, then GDB also might not be helpful to execute the code and access the 'buffer' of the code to see the content of the flag.txt file.

Hence least privilege principle can be good risk mitigation for information disclosure vulnerability.