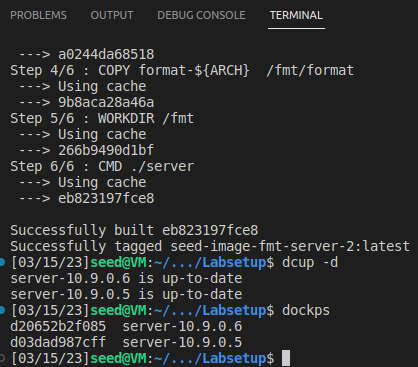
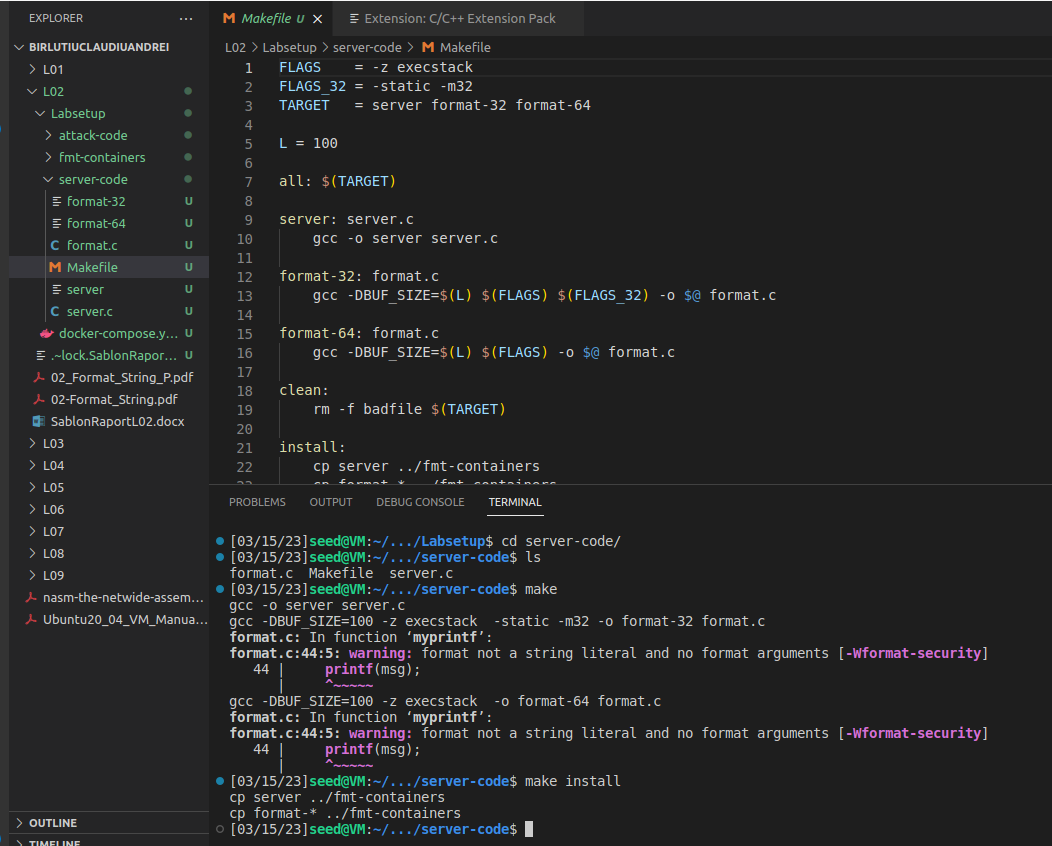
# Raport pentru lucrarea 2: Atacarea șirului format

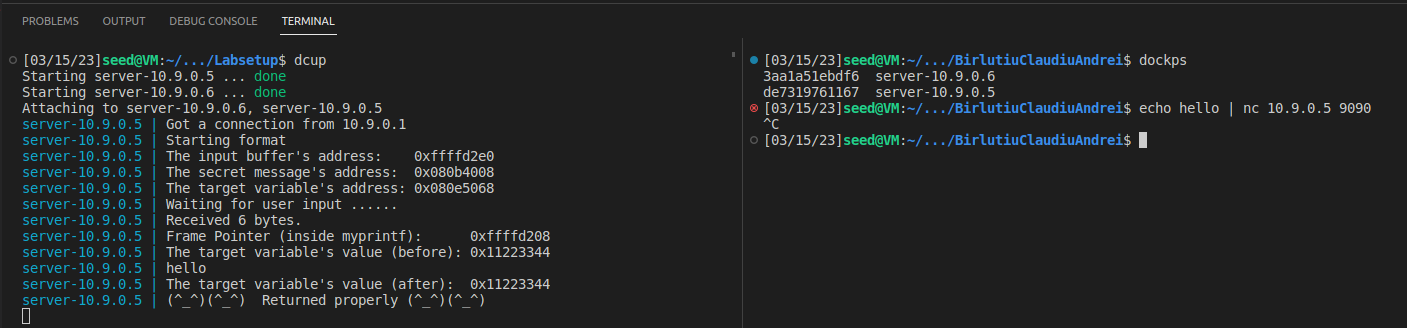
Autor: Birlutiu Claudiu-Andrei

## Sarcina 1: Eșuarea programului

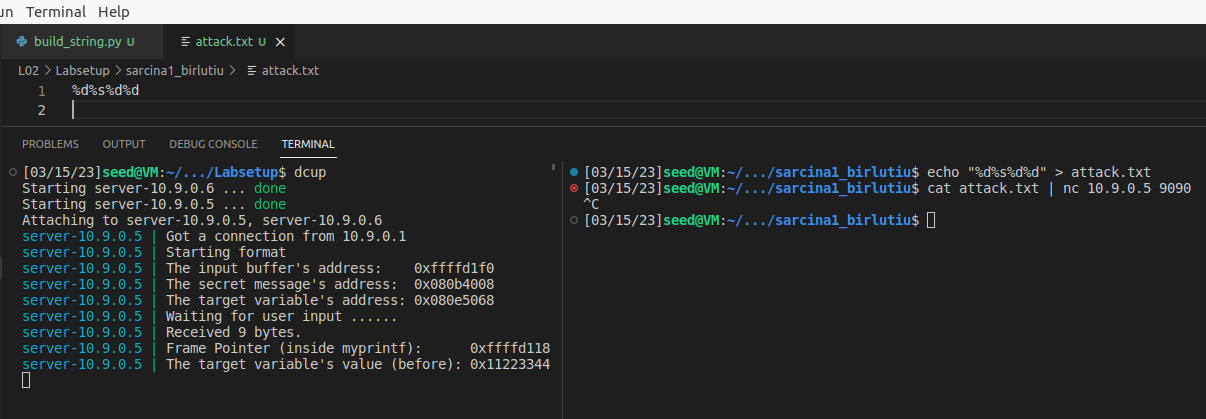
* În prima faza am rulat comenzile amintite în lucrearea de laborator pentru crearea serverului și lansarea lui în execuție în cele doua containere

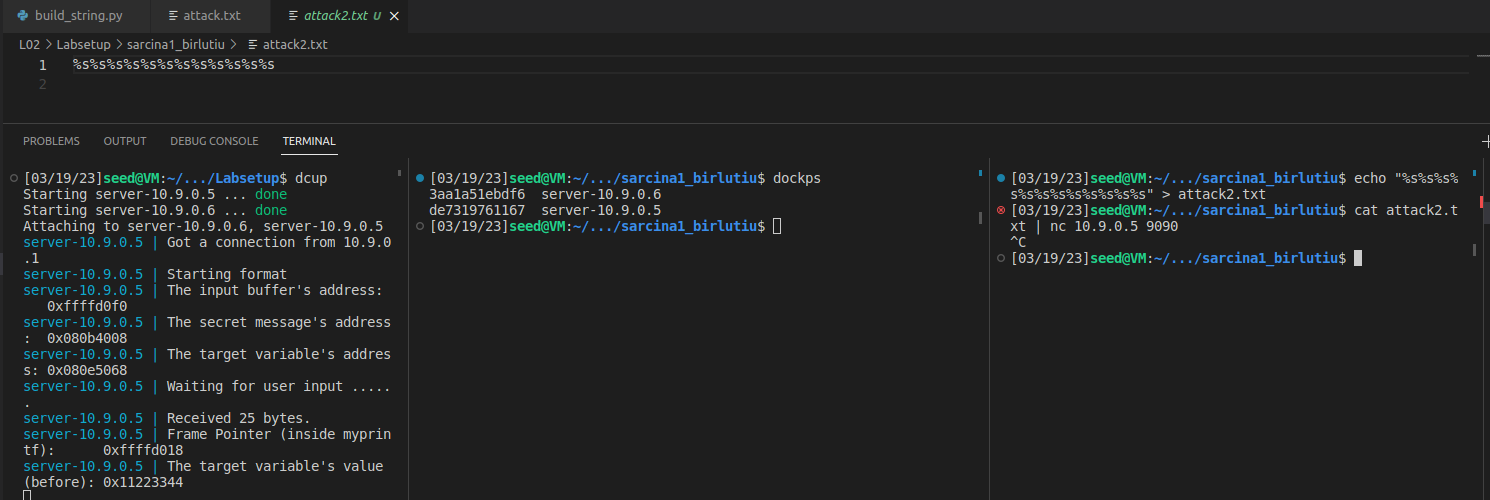


* testarea exemplului descris în laborator, prin care se conecteaza laserverulul 10.9.0.5 și trimitem ca input „hello”; serverul va afisa la stdout valoarea trimisa de noi



* atacul la nivelul codului prin inserarea unui string de tipul „%d%s%d%d” prin care se va accesa o zona invalida de pe stiva => duce la craparea; -la procent %s printf va accesa o zona invalida de memorie deaorece va trata valoare de pe stiva ce adresa; programului și nu se va afisa fetele zambarete; sau %s%s%s%s%s%s%s%s%s%s%s%s – programul va crapa deoarece pentru fiecare %s, functia va trata valoarea ca adresa și extrage date de la adresa respectiva => una dintre valorile de la acele adresa fiind invalida va duce la craprea programului
* programul server nu s-a blocat, programul **format** care a esuat ruleaza intr-un proces copil generat de programul server

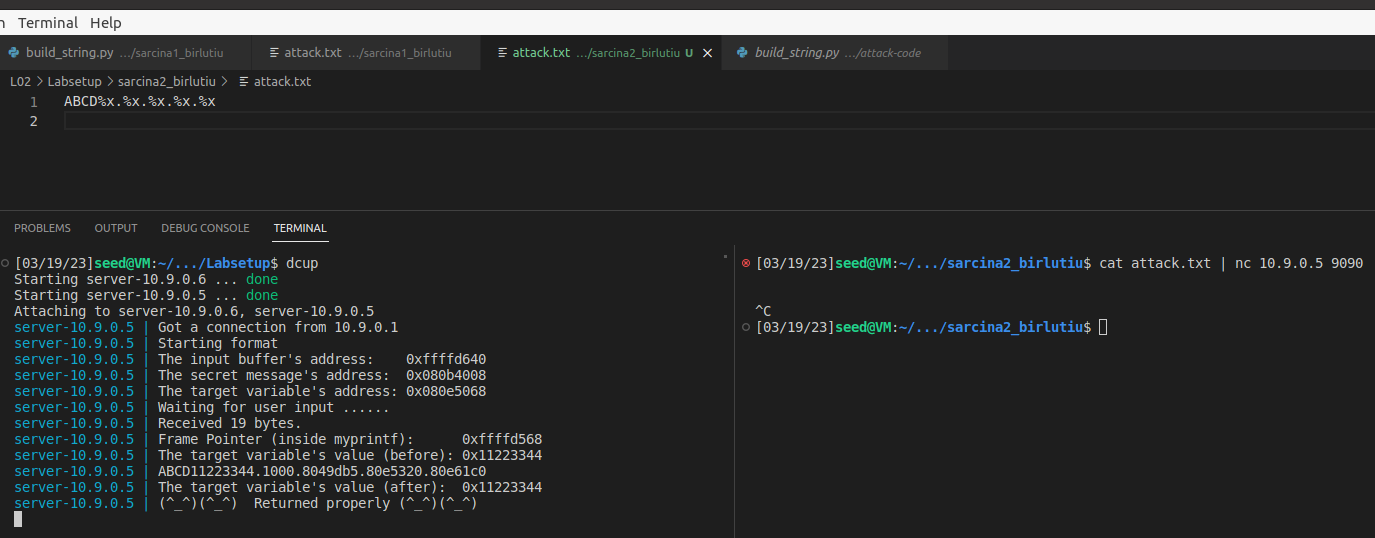




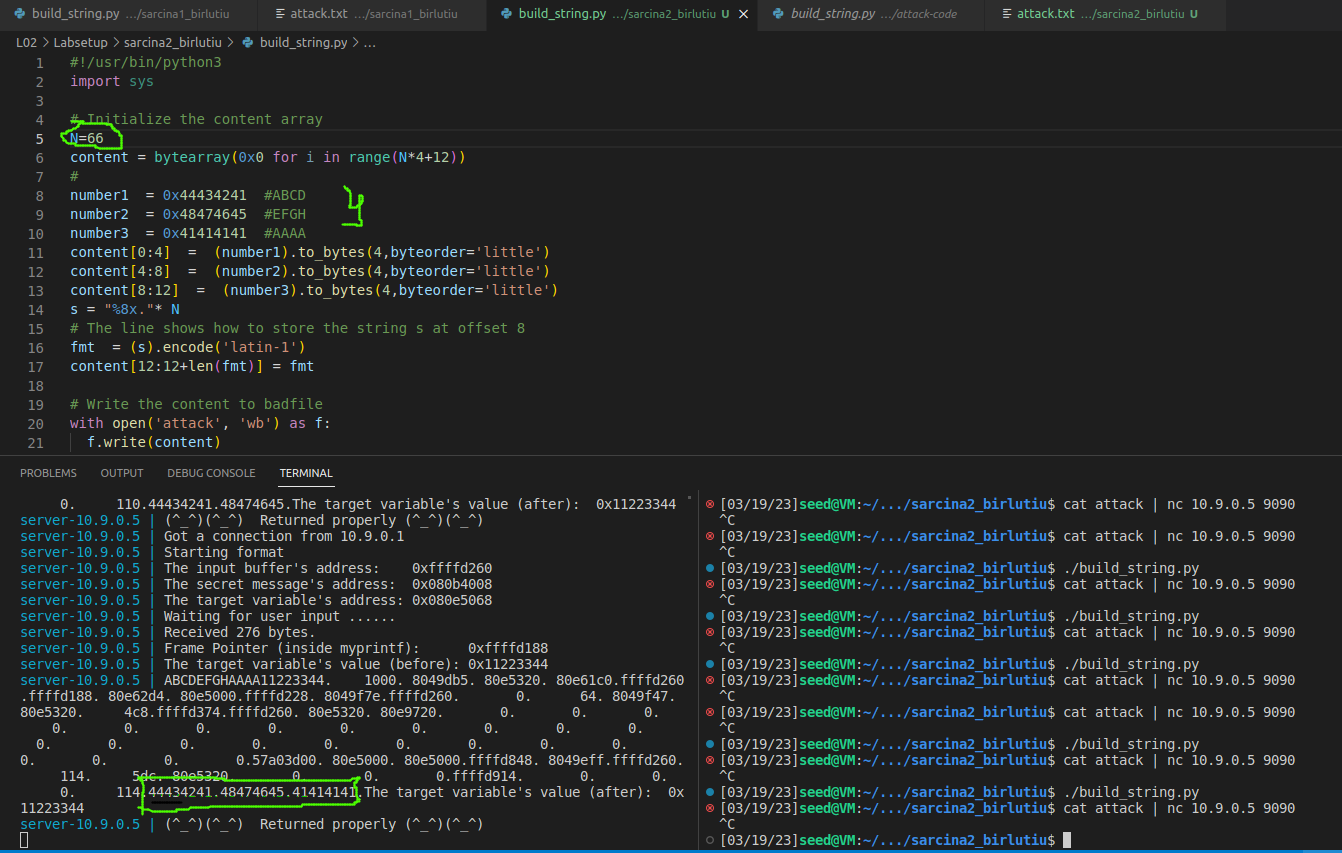
## Sarcina 2: Tipărirea conținutului memoriei serverului

### Sarcina 2.A: Datele din stivă.

* Pentru afisarea continutului de pe stiva se va folosi un specificatorul **%x** care afiesaza continutul in hexazecimal. Se introduc astfel de speficatori in inputul de la user pentru afisarea unor date de pe stiva care pot fi secrete;
* in aceasta metoda, valorile sunt afisate de server, si nu sunt returnate spre clientul rau intentionat

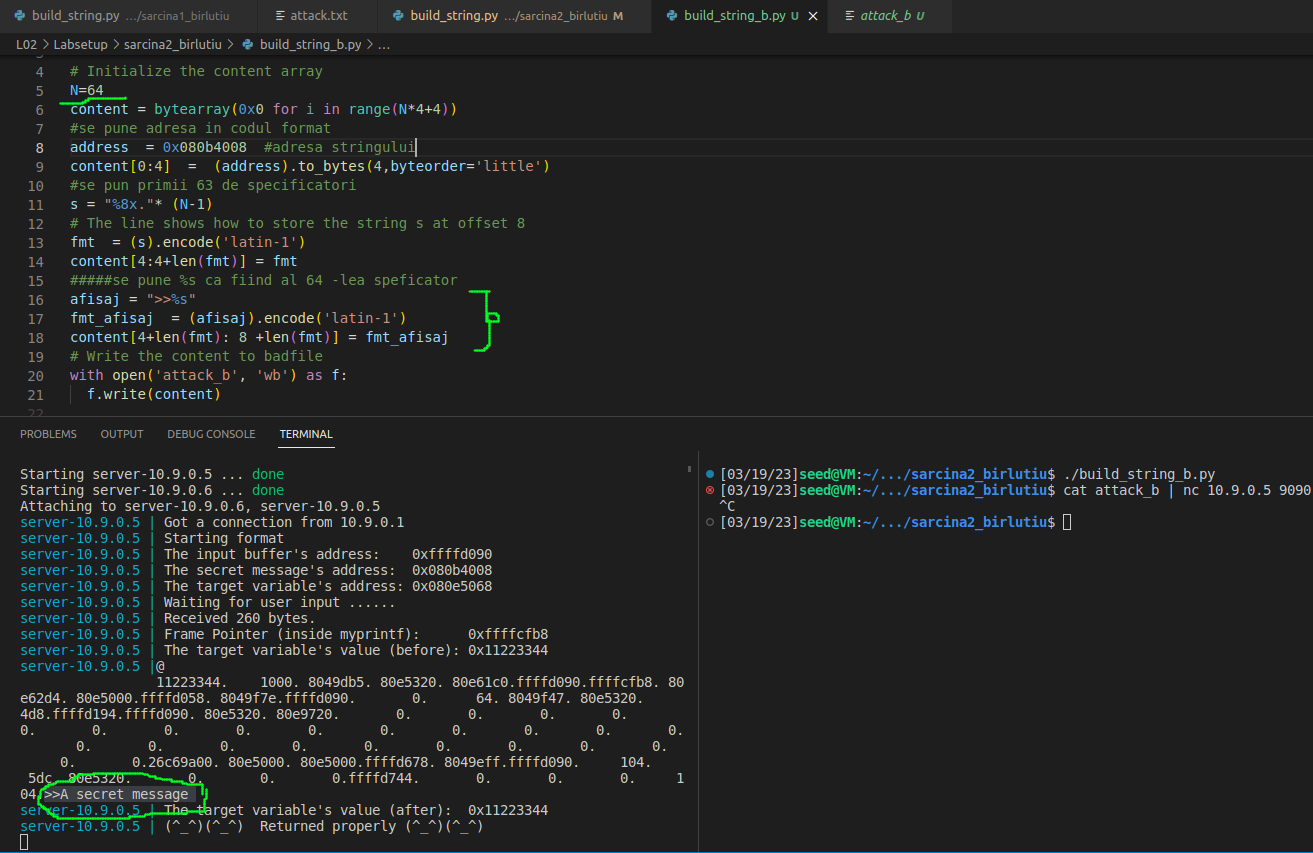


* in codul inserat se pot observa specificatori de form %x prin => iar ca rezulat se pot observa valorile de pe stiva ce sunt afisate prin iserarea unor astfel de specificatori → valoarea target = 0x11223344; fiind chiar capturata
* provocarea lansata in cadrul acestei sarcini se refera la afisarea primilor 4 octeti transmisi ca input; Vom lua 3 numere cu valoarea pentru a fi mai usor de a identifica seceventa de octeti. Scopul nostru e de identifica number1 (are 4 octeti)
  + - number1 = 0x44434241 #ABCD
    - number2 = 0x48474645 #EFGH
    - number3 = 0x41414141 #AAAA
* ne folosim de o varianta a programului build\_string.py pentru a forma mai multe tipuri de formate. Numărul de **%x** necesari se decide pe baza distanței dintre valoarea de început a pointerului de argumente și variabilă. Numărul de %x se obține prin încercări
* in programul de **build\_string.py** N reprezinta numarulu de %8x folositi ca specificatori, iar 3 siruri de octeti sunt furnizati ca intrare la inceput pentru a observa cu exactitate ca s-a ajuns pe stiva la valoarea furnizata ca input; 8 – inseamna ca se padding prin spatiere;
* in urma mai multor incercari s-a ajuns la concluzia ca avem nevoie de 66 de specificatori pentru a afisa cei 12 octeti (ABCDEFGHAAAA). Ceea ce inseamna ca pentru afisarea primilor 4 octeti este nevoie de **64** de specificatori %8x



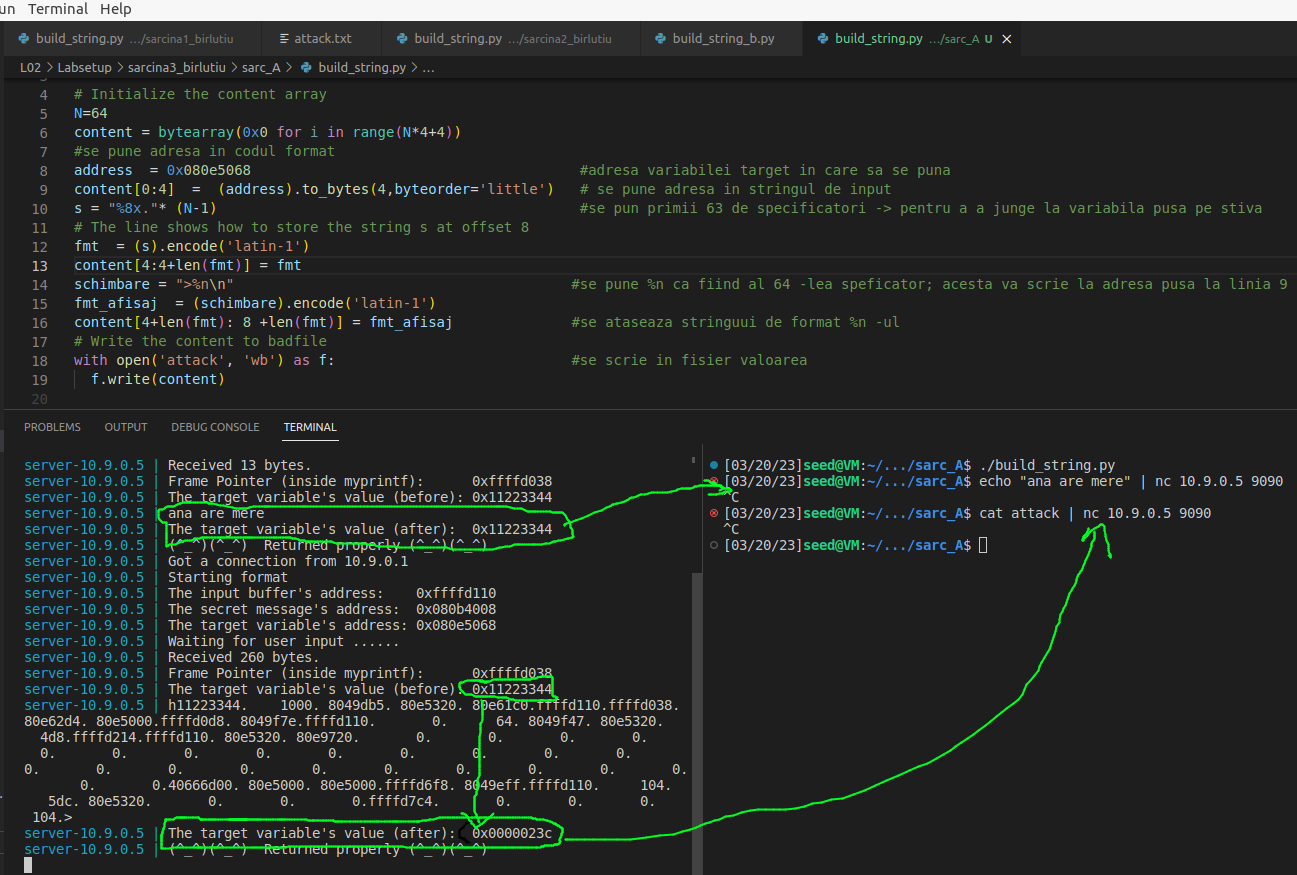
### Sarcina 2.B: Datele din heap.

* In informatiile furnizate de catre server, acesta mentioneaza faptul ca adresa unde se afla secret-ul este 0x080b4008
* punem adresa in formatul de string pentru printf, iar apoi al 64-lea speficator de format va fi %s, care va merge si va citi valoarea de la adresa data la inceput



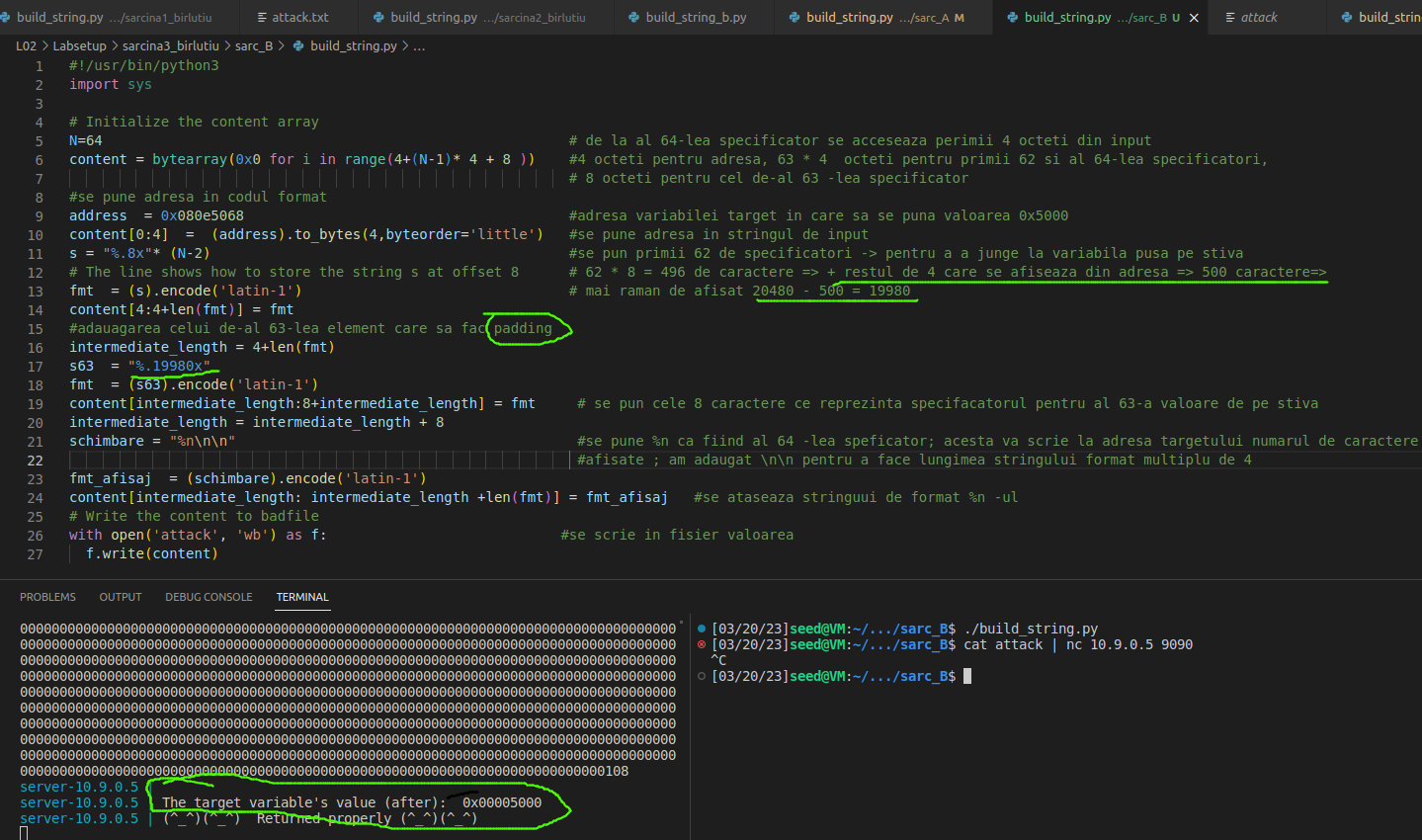
## Sarcina 3: Modificarea memoriei programului server

### Sarcina 3A. Schimbarea valorii la o valoare diferită

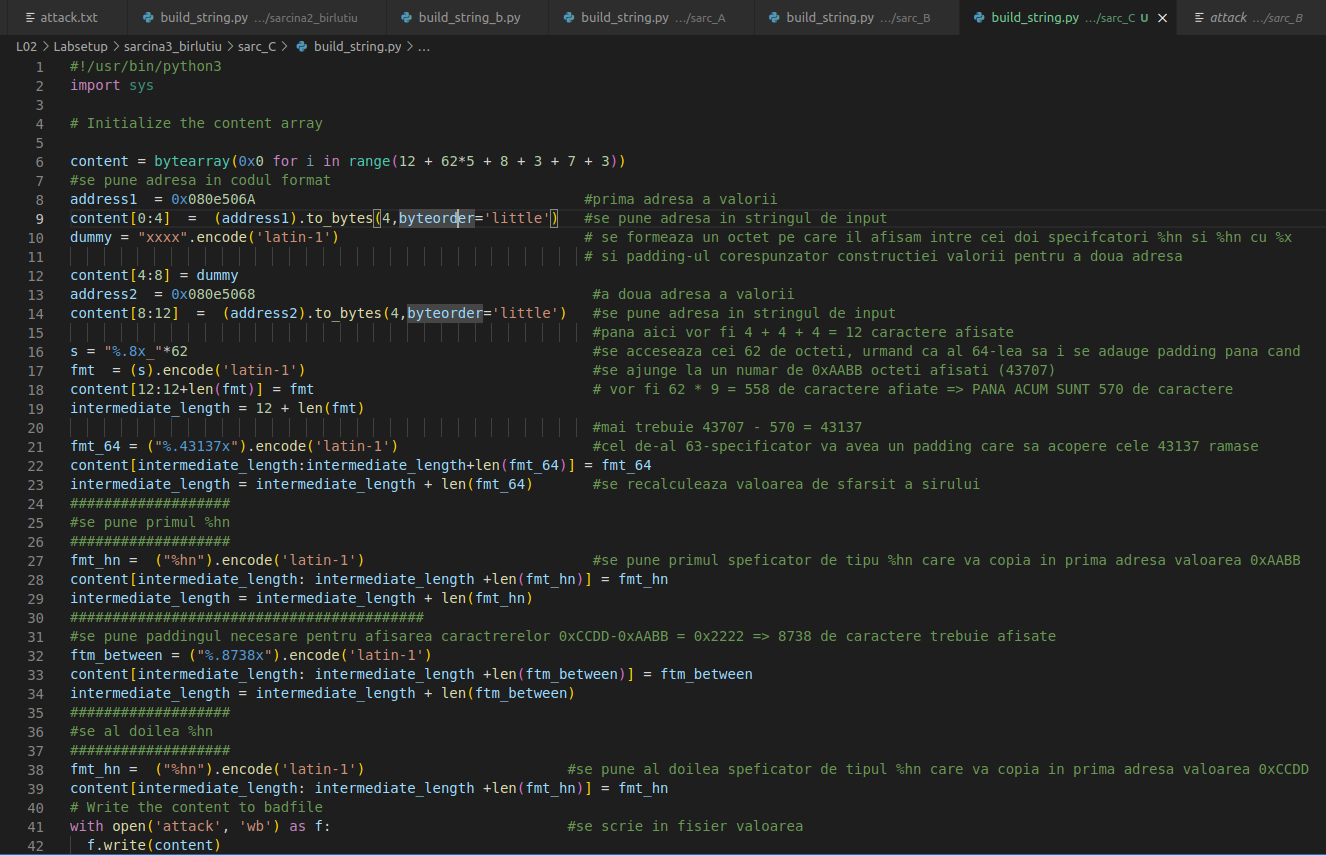


### Sarcina 3A. Schimbarea valorii la 0x500

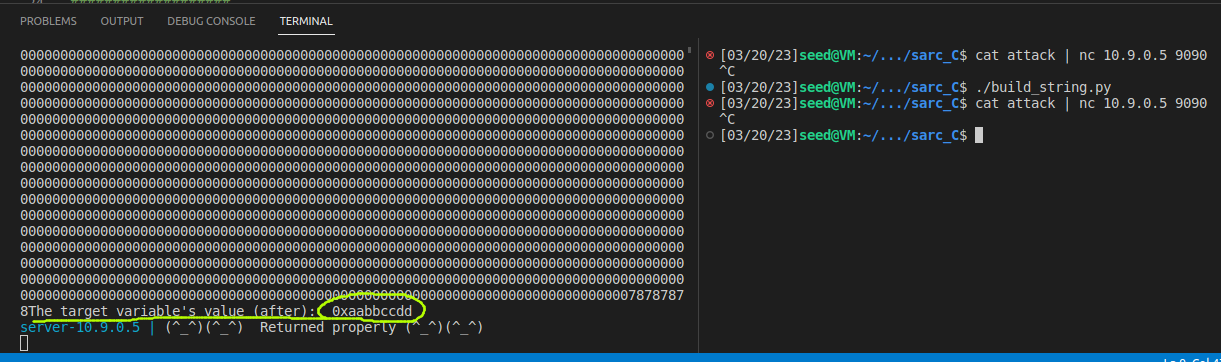
* Se observa ca adresa targeturlui este afisata la stdout din 2 caractere: h și spațiu liber: 5000 în baza 16 reprezninta 20480 în decimal. Deci, înainte de specificatorul %n trebuie afisate 20480 de caractere pentru a fi stocat în variabila target valoarea 5000 în baza 16 (20480). Pentru ca doua caractere sunt consumate de afisarea adresei, va trebui să facem padding la afisarea în hexa pentru a afisa restul de 20478 de caractere.
* Construim un build\_string.py care sa construiască input-ul necesar acestei operatii; STIIM ca al 64 specificator face referire la variabila încărcata pe stiva, în cazul nostru adresa variabile target; %n- va scrie la adresa aceasta valoarea caracterelor afisate pana la întâlnirea specificatorului; ca idee-> vom accesa primele 62 de valori de pe stiva cu specificatorul „**%.8x**”, urmând apoi a 63-a valoare la care ii facem padding cu numărul de caractere ramase pana la atingerea numarului de 20480 (0x5000)



### Sarcina 3A. Schimbarea valorii la 0xAABBCCDD



* Am obținut valoarea 0xAABBCCDD în target



## Sarcina 4: Obținerea unui shell invers

## Sarcina 5: Atacarea codului pe 64 biți

## Sarcina 6: Remedierea problemei