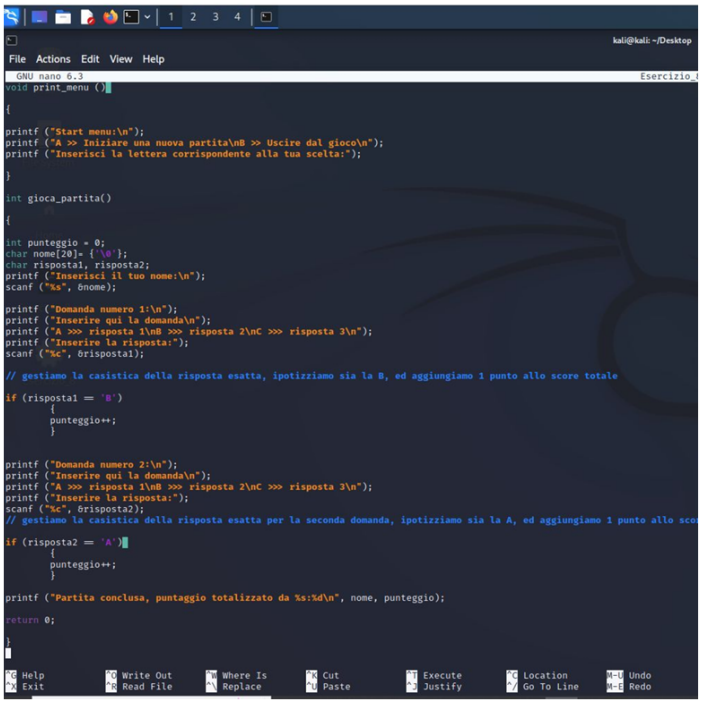
Esercizio S2-L4



Questo programma avvia un semplice gioco a domande che inizia con la richiesta di selezionare A o B e del tuo nome per poter avviare il gioco. L'iniziale richiesta di selezione viene eseguita impostando un semplice comando if e il comando while,dove se selezioni digitando A il gioco si avvia (condizione positiva impostata con while) e si bloccherà digitando B (condizione negativa che rende falsa la condizione impostata con if).
Successivamente ci viene richiesto di inserire il proprio nome. In questo caso il nome dovrà avere un numero di caratteri pari o inferiore a 20 come richiesto dall'array inserito in "char nome[20] = {'\0'};". Il char in questo comando è a variabile dove il carattere corrisponderà massimo a 1 byte.
Rispettando queste regole il gioco avrà inizio e sempre con delle condizioni di If che andranno ad assegnare un punteggio in base alla risposta data e poi andare a terminare il processo riportandoci nome e punteggio conquistato.
Andiamo ora ad analizzare i problemi e le complicità indesiderate che si verificano eseguendo il codice. Primo su tutti il programma si blocca riportando l'errore "segmentation fault", quindi il programma tenta di accedere alla memoria ma sarà in quanto bloccato perché non avrà il permesso di farlo. Questo avviene perché la memoria RAM sarà in crash a causa di una condizione di stack-overflow, questo avviene perchè la variabile impostata dovrebbe occupare uno spazio,che per convenzione chiameremo slot, pari ad un byte. Invece noi andando a inserire un nome con più di 20 caratteri andremo a invadere anche gli slot successivi che dovrebbero essere dedicati alle altre variabili. Questo problema si verifica a livello dello stack della ram, una delle componenti del buffer dal carattere dinamico dove vengono processate le informazioni che opera assieme al processore. Lo stack agisce con un sistema di Push e Pop definito come LIFO. Durante la fase di Push i dati vengono aggiunti allo stack mentre durante la fase di Pop i dati vengono rimossi (gli ultimi file aggiunti sono i primi ad essere rimossi). Un errore di overflowing in questo sistema può essere volontario o dato da un errore logico/battitura come nel nostro caso, ma anche intenzionale a causa di un attacco. In quest'ultima casistica, l'attaccante può inserire il malware in vari

slot suddiviso in piccole porzioni che verranno pian piano processati dalla CPU fino ad ottenere il malware intero. In altri casi viene iniettato nella RAM tramite l'arrow (iniettori di memoria) a seguito di un'involontaria autorizzazione dell'utente.
L'altra casistica di errore si verifica sempre in fase iniziale quando andiamo a impostare A o B per avviare o chiudere il gioco. Nel momento in cui noi digitiamo una lettera o carattere differente ad A o B il programma termina e si chiude nonostante, dalla lettura del programma, risulta la volontà di continuare la richiesta, data dall'impostazione di "return 0". Questo avviene perché abbiamo inserito il comando "void", che indica il ritorno di nessun valore.