TRACCIA 3: BWII

System Exploit BOF

Traccia Giorno 3:

Leggete attentamente il programma in allegato.

Viene richiesto di:

• Descrivere il funzionamento del programma prima dell’esecuzione

• Riprodurre ed eseguire il programma nel laboratorio

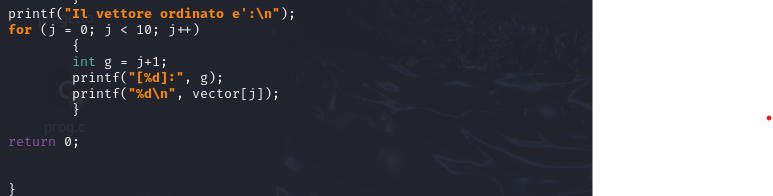
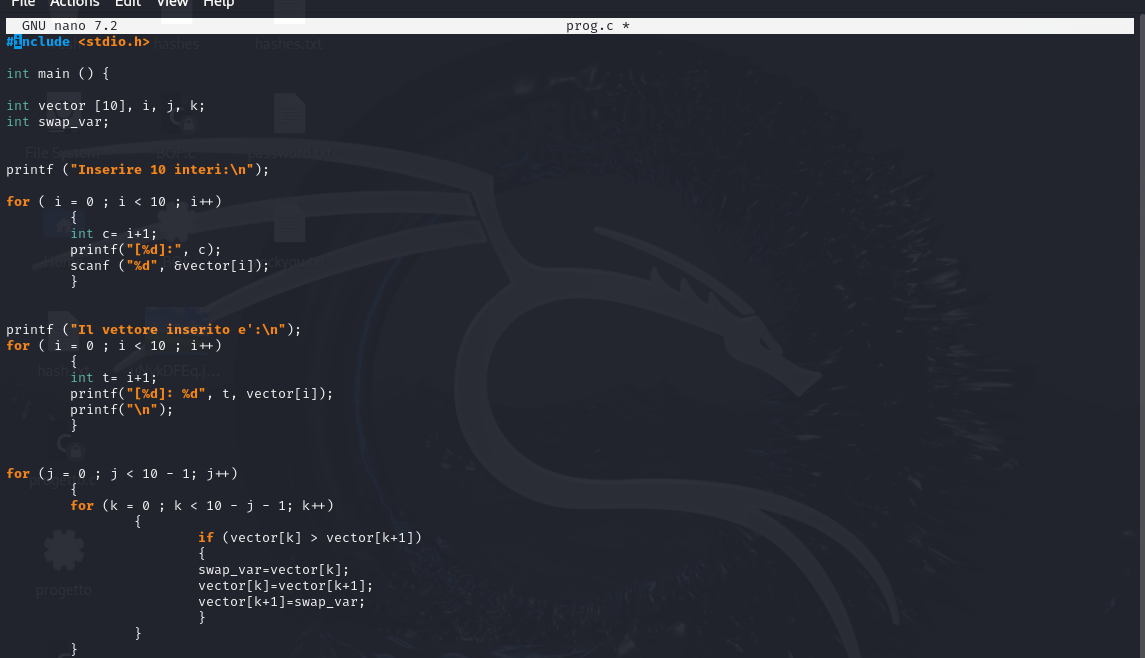
- le vostre ipotesi sul funzionamento erano corrette?

• Modificare il programma affinché si verifichi un errore di segmentazione

Suggerimento: Ricordate che un BOF sfrutta una vulnerabilità nel codice relativo alla mancanza di controllo dell’input utente rispetto alla capienza del vettore di destinazione.

Concentratevi quindi per trovare la soluzione nel punto dove l’utente può inserire valori in input, e modificate il programma in modo tale che l’utente riesca ad inserire più valori di quelli previsti.

GUIDA:

Spiegazione codice:

-#include<stdio.h>: Include la libreria input/output che fornisce per esempio le funzioni printf e scanf

-int main(): Dichiarazione della funzione main, punto di ingresso del programma

-int vector[10]: Un array di 10 elementi specificato dalle variabili: i, j, k

-Int swap\_var: Una variabile per lo scambio dei valori durante l’ordinamento del vettore

-printf: Stampa un messaggio di richiesta di inserire 10 elementi da parte dell’utente

-Ciclo For: Permette all’utente di inserire 10 interi e li memorizza nell’array vector, dove:

Int c = 1 + i: È utilizzata per mostrare l'indice dell'elemento attuale dell'array all'utente in un formato più comprensibile. viene impostata uguale ad i + 1, poiché gli indici degli array in C iniziano da 0, ma potrebbe essere più intuitivo per l'utente se iniziano da 1.

Printf(“[%d], c”): Stampa l'indice dell'elemento attuale dell'array all'utente.

Scanf(“%d”, &vector[i]): Legge un intero inserito dall'utente e lo memorizza nell'array

-Ciclo for (successivo 1): Serve per stampare il vettore inserito

-Algoritmo di ordinamento a bolle:

for (j = 0; j < 10 - 1; j++)

{for (k = 0; k < 10 - j - 1; k++)

{if (vector[k] > vector [k + 1])

{

// Scambio gli elementi se sono in ordine errato

swap\_var = vector[k]; vector[k] = vector [k + 1];

vector [k + 1] = swap\_var;

È un algoritmo che funziona confrontando ripetutamente le coppie dei valori vicini e li scambia se sono in ordine errato ovvero, si inizia dal primo elemento e lo confronta con il successivo, se nota che l’elemento corrente è maggiore del successivo li scambia, continuando così fino alla fine dell’array escludendo l’ultimo elemento già ordinato

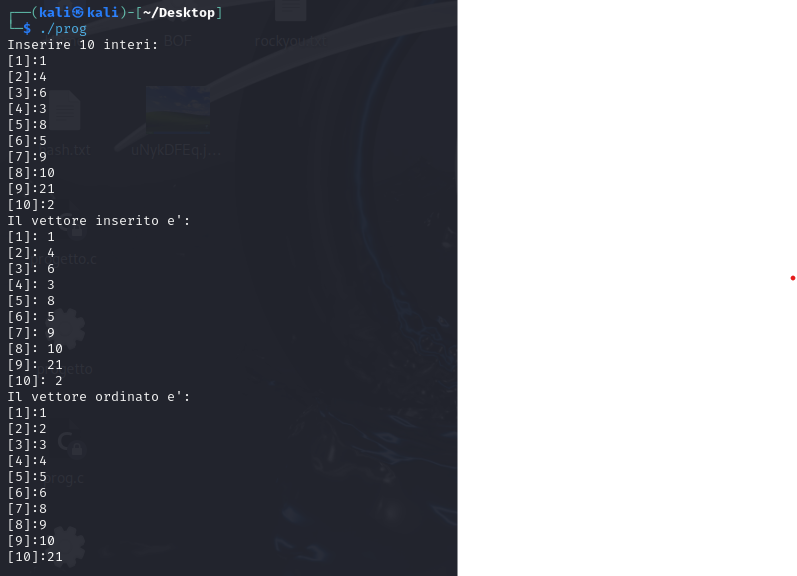
-Ciclo for (successivo 2): Serve per stampare il vettore ordinato.

-return 0: Termina il programma

Per far sì che si crei un errore di segmentazione basterebbe semplicemente abbassare il numero di elementi dell’array, facendolo passare per esempio da 10 a 5 banalmente, e aumentato il numero di elementi inseribili da 10 a 15 (vedere le immagini successive)

Il programma cercherà comunque di inserire i 15 elementi considerando il ciclo for, e di conseguenza si creerà un errore di segmentazione sovrascrivendo la memoria oltre i limiti dell’array. Vediamo:

prog.c eseguito normalmente:

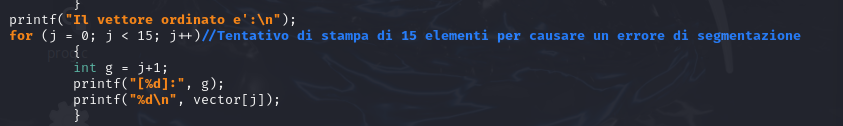


prog.c con il cambio di elementi dell’array:





(+ di 15. C’è un errore nei commenti)



BONUS:

