Report

Filip S.
CS, Week 2, Day 5

Bug Hunting

Introduzione:

Per bug hunting si riferisce la analizzazione dei software e hardware a caccia di vulnerabilità.

Obbiettivi:

1. Capire cosa fa il programma senza eseguirlo.
2. Individuare dal codice sorgente le casistiche non standard.
3. Individuare eventuali errori
4. Suggerimenti

Tools:

1. Virtual Lab (Kali)
2. GNU Nano 6.3

Indice

1.Capire cosa fa il programma senza eseguirlo

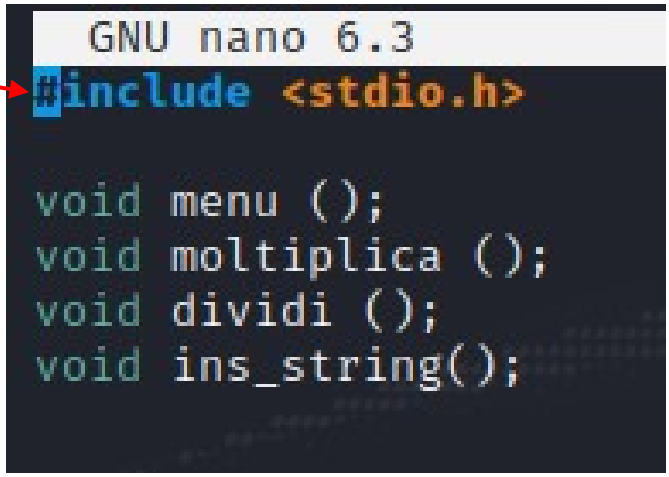
la libreria standard input-output header

funzione menu = suggerisce un menu del introduzione al utente

funzione moltiplica = suggerisce che ci siano delle moltiplicazioni

funzione dividi = suggerisce che ci siano delle divisioni

funzione ins_string = suggerisce (insert_string) quindi input dal utente del testo però non è ancora chiaro per cosa.



1.1 La funzione principale <void main>

variabile scelta = suggerisce un input da parte del utente

switch scelta = suggerisce che avremo piu di una condizione che utente puo scegliere

case A = per scegliere la moltiplicazione

case B = per scegliere la divisione

case C = per inserire la string ? Non è ancora chiara la questa funzione.

1.2 La prossima funzione è <void menu>

possiamo vedere la introduzione al utente, abbiamo tre “printf” che mettono in output il contenuto sul monitor.
Possiamo vedere dal contenuto che il programma dovrebbe infatti moltiplicare, dividere e inserire la stringa?, questa ultima non è ancora chiara.

```
void menu ()
{
        printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigarealcuni compiti\n");
        printf ("Come posso aiutarti?\n");
        printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >>Inserire una stringa\n");

}
```

1.3 La prossima funzione è <void moltiplica>

qui possiamo vedere:

le due variabili, a e b

output del testo che indica di fare input di due numeri

var. A del tipo FLOAT
var. B del tipo INT

la formula della moltiplicazione

output del calcolo

```
void moltiplica ()
{
        short int a,b = 0;
        printf ("Inserisci i due numeri da moltiplicare:");
        scanf ("%f", &a);
        scanf ("%d", &b);

        short int prodotto = a * b;

        printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}
```

1.4 La prossima funzione è <void dividi>

qui ci viene il dubbio se la funzione dovrebbe dividere o dare il resto della divisione

i suggerimenti che questa sia la divisione:
nome della funzione

indicazione di inserire il numeratore e denominatore

output del testo

il suggerimento che ci dia il resto della divisione:

```
void dividi ()
{
        int a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf ("Inserisci il denumeratore:");
        scanf ("%d", &b);

        int divisione = a % b;

        printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
```

la formula: a % b
Il maggior numero di suggerimenti ci suggerisce che sia una divisione

1.5 La prossima funzione è <void ins_string>

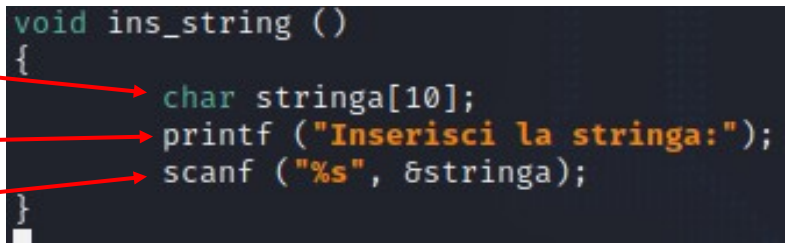
finalmente possiamo guardare nella funzione ’’ins_string’’ per capire il suo funzionamento

abbiamo la variabile char stringa con array di 10 byte = 9 caratteri

indicazione di inserire la stringa

input per utente della string

e nessun output dopo l’input del utente, che ci suggerisce che questa funzione non abbia senso logico.



2. Individuare dal codice sorgente le casistiche non standard

I problemi rilevati:

1. %d (sta per INT) >>> %c (sta per CHAR)

Visto che nella funzione <void menu> abbiamo da scegliere tra A, B e C dobbiamo cambiare il tipo di variabile in input.

2. la parentesi graffa aperta deve trovarsi accanto o sotto il int main(), in questo caso dovrebbe darci errore di sintassi, come se non esistesse la {

3. switch >>> ciclo if-else

Visto pocchi numeri di condizioni, è proponibile usare gli IF-ELSE statement; il switch è meglio usato quando si hanno tante condizioni. In questo caso sarebbe da aggiungere il “default” case, che potrebbe entrare in condizione se l’utente non inserisce uno dei caratteri indicati.

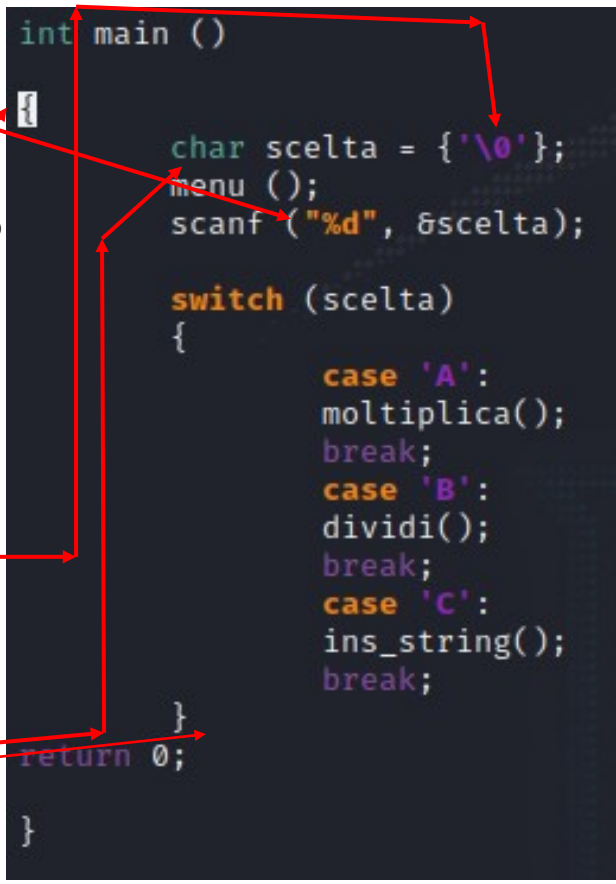
4. {‘\0’} >>> ‘\0’

Non servono le {}, le stesse vengono usate per il blocco di codice con piu di 1 statement

5. loop

Per far in modo che utente non debba lanciare il programma se sbaglia la lettera della var. scelta, sarebbe da creare un loop, semplicemente aggiungendo un “do”

e un “while”



3. Individuare eventuali errori

Errori

scanf(“%f”, &a); > scanf(“%d”);

se non viene cambiato il tipo del input della variabile, il risultato sarà sempre 0, perché il programma si aspetta un INT mentre input è assegnato come FLOAT.

```
void moltiplica ()
{
        short int a,b = 0;
        printf ("Inserisci i due numeri da moltiplicare:");
        scanf ("%f", &a);
        scanf ("%d", &b);

        short int prodotto = a * b;

        printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}
```

denumeratore > denominatore

% >> /
se non sostituiamo il “modulo” con “divisione” otterremo il risultato inaspettato

```
void dividi ()
{
        int a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf ("Inserisci il denumeratore:");
        scanf ("%d", &b);

        int divisione = a % b;

        printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
```

% = modulo
modulo serve per restituirci il resto della divisione.

/ = divisione

4. Suggerimenti

4.1 [Importante] Rimuovere la funzione <void ins_string>

La funzione non ha nessuna ragione di essere presente nel codice, è soltanto una vulnerabilità di overflow del buffer.

4.2 [Importante] Per la variabile “scelta”

Dentro la funzione <int main> dopo aver aggiunto while (pag.4 punto 2) per i motivi di sicurezza si consiglia di assegnare la condizione di controllo del input se utente inserisce una lettera diversa da quelle istruite. Esempio: “while(scelta != ‘A’);”

4.3 Nella funzione <void dividi>

Sostituire input del tipo delle variabili “a”, “b” e “divisione” dal INT nel FLOAT per avere i numeri reali nel input e nel output, visto che nelle divisioni c’è più possibilità di usare i numeri decimali.

4.4 Commenti

Si consiglia sempre di scrivere dei commenti durante la programmazione con delle spiegazioni usando (//, /* e */) per aiutare ad altri programmatori a capire il proprio codice.

05/11/2022

Filip S.