Progetto S2L5 Bug Hunting

Per agire come un Hacker bisogna capire come pensare fuori dagli schemi. L'esercizio di oggi ha lo scopo di allenare l'osservazione critica. Dato il codice in allegato, si richiede allo studente di:

1. Capire cosa fa il programma senza eseguirlo (vedi codice sulle ultime pagine)
2. Individuare dal codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati)
3. Individuare eventuali errori di sintassi / logici-Proporre una soluzione per ognuno di essi

1) Il programma è un semplice assistente digitale che fornisce all'utente un menu con tre opzioni (A, B, C) e svolge compiti diversi in base alla scelta dell'utente.

1. Opzione A Moltiplicazione: viene chiamata la funzione **moltiplica()**. Questa funzione chiede all'utente di inserire due numeri interi e quindi calcola il loro prodotto, mostrando il risultato.
2. Opzione B Divisione: viene chiamata la funzione **dividi()**. Questa funzione chiede all'utente di inserire due numeri interi (numeratore e denominatore) e quindi calcola il risultato della divisione, mostrando il risultato.
3. Opzione C inserimento di una Stringa: viene chiamata la funzione **ins_string()**. Questa funzione chiede all'utente di inserire una stringa

2) Il codice presenta qualche problema potenziale: 1) nella scelta inziale in caso l'utente non vada ad inserire le richieste specifiche delle opzioni del programma per esempio se io in fase di scelta A, B, o C vado ad inserire D o un altro carattere il programma smette di funzionare perché non è stata prevista la possibilità che l'utente possa non fare ciò che gli viene richiesto nelle varie fasi di selezione delle opzioni, un'alternativa a questo problema sarebbe stato mettere una funzione in grado di riconoscere il comando sbagliato o se il comando non è tra le varie opzioni mandi un messaggio su schermo che dica all'utente che l'opzione prevista non esiste e che deve scegliere solo tra quelle previste. 2) Un altro possibile problema si va a creare quando l'utente va ad inserire numeri reali per lo svolgimento dei calcoli, in tal caso il programma non darebbe un risultato coretto, una possibile soluzione sarebbe stato inserire la variabile float invece che int così da poter far svolgere i calcoli richiesti con qualsiasi numero. 3) La funzione che permette di inserire una stringa presenta un problema di buffer overflow che potenzialmente può causare il crash dell'applicazione o essere usata come vulnerabilità dall'attaccante per inserire pezzi di codice malevolo frammentati. In conclusione, questo programma necessita di bug fixing prima di poter essere utilizzato dall'utente finale.

3) Nel codice analizzato ho riscontrato 3 diversi bug e un possibile buffer overflow:

1. **Errore nella funzione switch**:
   char scelta = {'\0'};

   menu ();

   scanf ("%d", &scelta);

   Nei casi nello switch ci sono caratteri ('A', 'B', 'C'), ma si sta usando %d per la lettura della scelta. Bisognerebbe usare %c per la lettura di un carattere, scanf (" %c", &scelta);

2. **Errore nella funzione di moltiplicazione**:

```
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);
}
```

   Si sta cercando di leggere un numero **float** (%f) in scanf ("%f", &a) e numero in in , scanf ("%d", &b);ma a è dichiarato come **short int** precedentemente. Bisogna usare %hd per la lettura dei numeri **short int**.

```
void moltiplica()
{
   short int a, b = 0;
   printf("Inserisci i due numeri da moltiplicare:");
   scanf("%hd", &a);
   scanf("%hd", &b);
   short int prodotto = a * b;
   printf("Il prodotto tra %hd e %hd e': %hd", a, b, prodotto);
}
```

3. **Errore nella funzione dividi**:
   int divisione = a % b;

   Si sta usando l'operatore modulo (%) per cercare di svolgere una divisione intera. Per ottenere il risultato della divisione, si usa l'operatore divisione (/), int divisione = a / b;

4. **Potenziale buffer overflow nella funzione di stringa**:

```
void ins_string ()
{
        char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", &stringa);
}
```

L'opzione stringa mette un limite massimo all'array di 10 caratteri, se l'utente mette un numero di caratteri maggiore dell'array, l'utente andrà a creare una situazione di buffer overflow che causerà un crash dell'applicazione. Per risolvere questo problema si può impostare la lunghezza massima della stringa in modo tale che l'utente non possa andare a sfruttare questa vulnerabilità, scanf("%9s", stringa);

Codice da analizzare

```
#include <stdio.h>

void menu ();
void moltiplica ();
void dividi ();
void ins_string();


int main ()

{
	char scelta = {'\0'};
	menu ();
	scanf ("%d", &scelta);

	switch (scelta)
	{
		case 'A':
		moltiplica();
		break;
		case 'B':
	  dividi();
	  break;
		case 'C':
	  ins_string();
	  break;
	}

return 0;

}


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

```
}


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);
}


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);
}


void ins_string ()
{
        char stringa[10];
    printf ("Inserisci la stringa:");
    scanf ("%s", &stringa);
}
```