Manuel Di Gangi

S7_L4

Buffer overflow

7 marzo 2024

INDICE

1. Traccia

Provare a riprodurre l'errore di segmentazione modificando il programma come di seguito:

- Aumentando la dimensione del vettore a 30;

2. Buffer Overflow

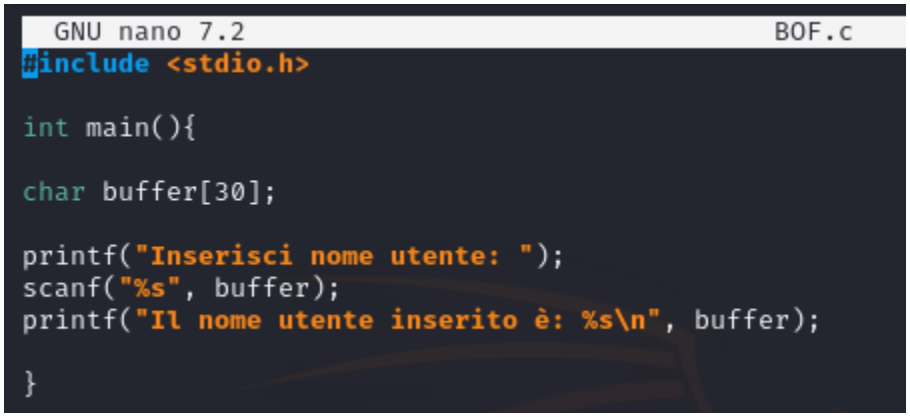
Il buffer overflow, o sovrafflusso del buffer, è una vulnerabilità di sicurezza che si verifica quando un programma o un processo tenta di scrivere dati oltre i limiti di un buffer di memoria allocato. Un buffer è una zona di memoria temporanea utilizzata per immagazzinare dati durante l'esecuzione di un programma. Se un programma non gestisce correttamente la quantità di dati che scrive in un buffer, può verificarsi un overflow, andando a sovrascrivere parti di memoria adiacenti.

Questa vulnerabilità è spesso sfruttata da attacchi informatici per inserire e eseguire codice dannoso nel sistema. Il processo di sfruttamento di un buffer overflow generalmente coinvolge il sovrascrivere il buffer con dati dannosi in modo da modificare il flusso di esecuzione del programma. Se un attaccante riesce a iniettare codice malevolo in un buffer e a farlo eseguire dal programma, può ottenere il controllo del sistema o eseguire azioni dannose.

Per prevenire i buffer overflow, è essenziale implementare buone pratiche di programmazione, come la validazione dei dati in ingresso, l'uso di funzioni sicure per la gestione dei buffer (come strcpy_s in C++) e l'uso di strumenti di analisi statica e dinamica per individuare potenziali vulnerabilità. Inoltre, molte piattaforme e compilatori offrono misure di sicurezza, come ASLR (Address Space Layout Randomization) e DEP (Data Execution Prevention), che possono contribuire a mitigare gli effetti di un buffer overflow

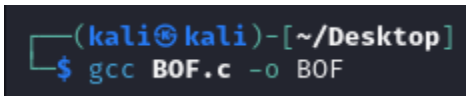
3. Il codice

Andiamo a scrivere un codice molto elementare che prende in input una stringa. Il codice è volutamente vulnerabile, in quanto non viene effettuata alcuna operazione di sanitizzazione dell'input. La stringa inserita dall'utente viene memorizzata nella variabile buffer. In C le stringhe vengono gestite come se fossero degli array di caratteri, nel nostro caso l'array ha dimensione 30, il che significa che può ospitare 29 caratteri inseriti dall'utente più il carattere di terminazione /0.

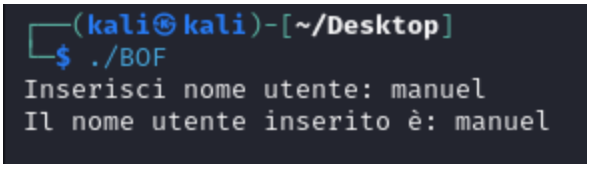


4.Esecuzione

Compiliamo il codice con il comando gcc e creiamo il file eseguibile:



Eseguiamo il codice inserendo un input "sicuro" per verificarne il funzionamento:



4

Eseguiamo nuovamente il codice, stavolta puntiamo a sfruttare la vulnerabilità inserendo più caratteri di quelli previsti.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Inserisci nome utente: masqwerflodifkdkfjfkdjdkfkdldldkjdkdjfkgkdfjfjgkfkfjgkfkfjgkfjfgj
Il nome utente inserito è: masqwerflodifkdkfjfkdjdkfkdldldkjdkdjfkgkdfjfjgkfkfjgkfkfjgkfjfgj
zsh: segmentation fault  ./BOF
```

Come possiamo vedere al momento del salvataggio in memoria della stringa il programma si interrompe segnalando un errore di ***Segmentation fault***