BUFFER OVERFLOW

CONSOLIDAMENTO CONOSCENZE

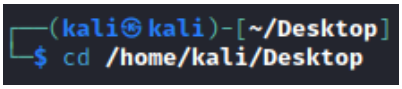
Traccia:
Nella lezione dedicata agli attacchi di sistema, abbiamo parlato dei buffer overflow, una vulnerabilità che è conseguenza di una mancanza di controllo dei limiti dei buffer che accettano input utente.
Nelle prossime slide vedremo un esempio di codice in C volutamente vulnerabile ai BOF, e come scatenare una situazione di errore particolare chiamata "segmentation fault", ovvero un errore di memoria che si presenta quando un programma cerca inavvertitamente di scrivere su una posizione di memoria dove non gli è permesso scrivere (come può essere ad esempio una posizione di memoria dedicata a funzioni del sistema operativo).

Riportare il codice che segue sulla macchina Kali Linux, creando un nuovo documento con estensione `.c` sul desktop

Per creare un nuovo documento su Kali, apriamo il terminale in alto a sinistra come mostrato in figura:

Dal terminale, ci spostiamo sul Desktop con il comando:

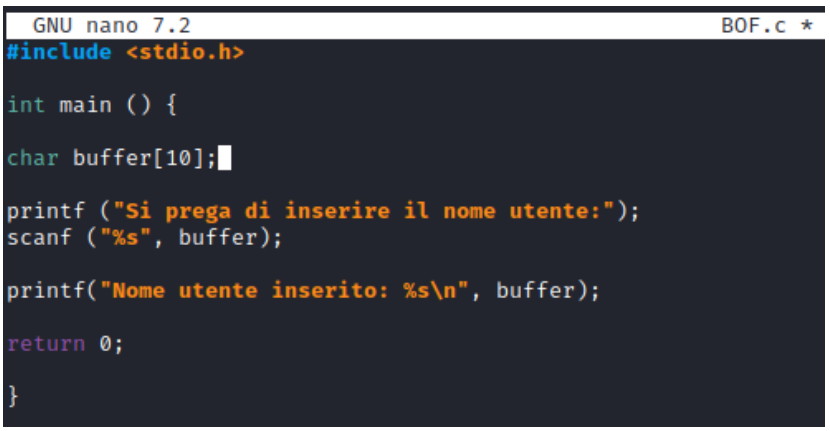
```
$ cd /home/kali/Desktop
```



Eseguiamo l'editor di testo nano, che ci permetto di aprire un file esistente, oppure di crearne uno nuovo se il nome del file specificato non esiste. Eseguiamo quindi il seguente comando per creare il file 'BOF.c'

```
$ nano BOF.c
```

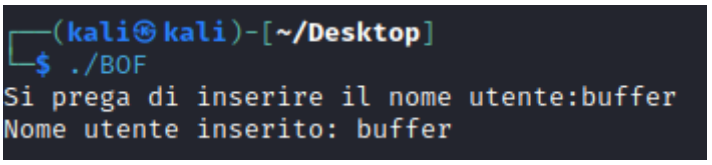
Riportiamo il frammento di codice:



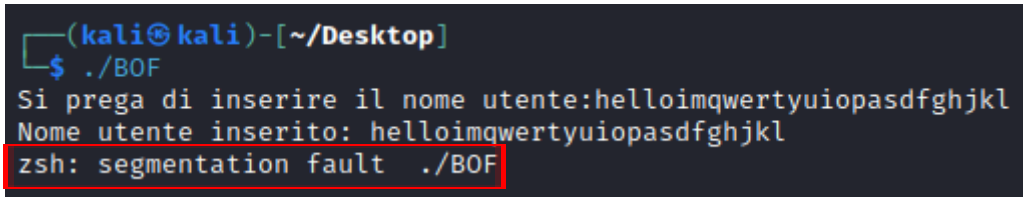
A questo punto, possiamo compilare il file utilizzando il comando

```
$ gcc -g BOF.c -o BOF
```

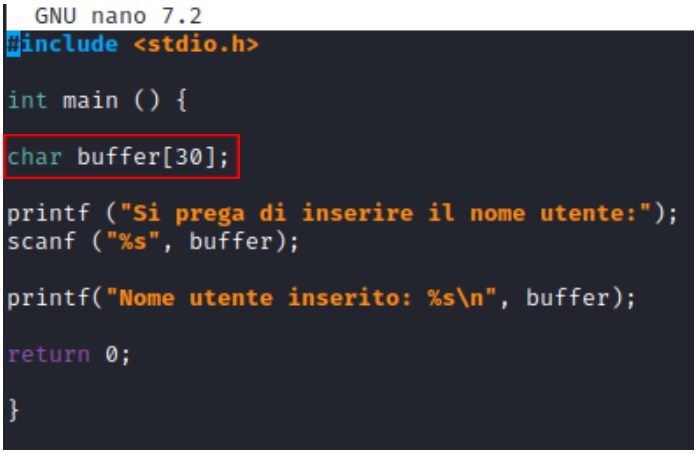
Inserendo un nome utente con 6 caratteri, il programma non ci riporta alcun tipo di problema, questo perché il buffer accetta fino a 10 caratteri.



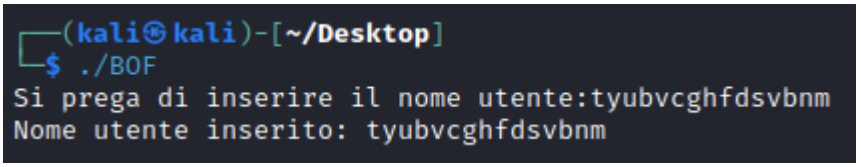
Infatti, se dovessimo inserire un nome utente con ad esempio 26 caratteri, ci ritorna un errore: 'segmentation fault'. L'errore di segmentazione infatti, avviene quando un programma, tenta di scrivere contenuti su una porzione di memoria alle quale non ha accesso. Questo è un chiaro esempio di BOF, abbiamo inserito 26 caratteri in un buffer che ne può contenere 10, di conseguenza alcuni caratteri stanno sovrascrivendo aree di memoria inaccessibili



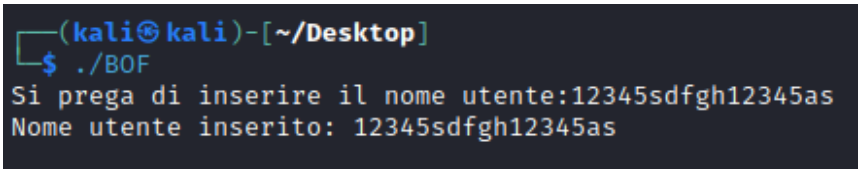
Aumentando il vettore a 30 e inserendo un nome utente da 30



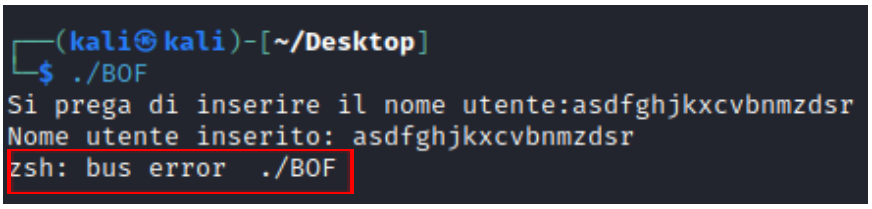
Da 15 caratteri:



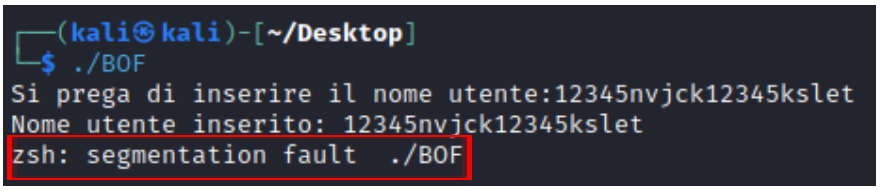
Da 17 caratteri:



Da 18 caratteri:



Da 20 caratteri:



ERRORE BUS

In informatica, un errore di bus è un errore generato dall'hardware, che notifica a un sistema operativo (OS) che un processo sta tentando di accedere alla memoria a cui la CPU non può indirizzare fisicamente. Nell'uso moderno sulla maggior parte delle architetture questi sono molto più rari degli errori di segmentazione, che si verificano principalmente a causa di violazioni dell'accesso alla memoria.

LE CAUSE

- indirizzo inesistente
  Il software indica alla CPU di leggere o scrivere un indirizzo di memoria fisica specifico. Di conseguenza, la CPU imposta questo indirizzo fisico sul proprio bus di indirizzi e richiede a tutti gli altri hardware collegati alla CPU di rispondere con i risultati, se rispondono per questo indirizzo specifico. Se nessun altro hardware risponde, la CPU genera un'eccezione, indicando che l'indirizzo fisico richiesto non è riconosciuto dall'intero sistema informatico. Si noti che questo riguarda solo gli indirizzi di memoria fisica. Il tentativo di accedere a un indirizzo di memoria virtuale non definito è generalmente considerato un errore di segmentazione piuttosto che un errore di bus, anche se la MMU è separata, il processore non può distinguere la differenza.

- accesso non allineato
  La maggior parte delle CPU sono indirizzabili in byte, dove ogni indirizzo di memoria univoco si riferisce a un byte a 8 bit. La maggior parte delle CPU può accedere a singoli byte da ciascun indirizzo di memoria, ma generalmente non può accedere a unità più grandi (16 bit, 32 bit, 64 bit e così via) senza che queste unità siano "allineate" a un limite specifico (la piattaforma x86 è un'eccezione notevole).

  Ad esempio, se gli accessi multibyte devono essere allineati a 16 bit, gli indirizzi (indicati in byte) a 0, 2, 4, 6 e così via sarebbero considerati allineati e quindi accessibili, mentre gli indirizzi 1, 3, 5 e così via sarebbero considerati non allineati. Allo stesso modo, se gli accessi multibyte devono essere allineati a 32 bit, gli indirizzi 0, 4, 8, 12 e così via sarebbero considerati allineati e quindi accessibili e tutti gli indirizzi intermedi sarebbero considerati non allineati.

  Le CPU generalmente accedono ai dati all'intera larghezza del loro bus dati in ogni momento. Per indirizzare i byte, accedono alla memoria all'intera larghezza del loro 'bus' dati, quindi mascherano e spostano per indirizzare il singolo byte. I sistemi tollerano questo algoritmo inefficiente, in quanto è una caratteristica essenziale per la maggior parte del software, in particolare l'elaborazione delle stringhe. A differenza dei byte, le unità più grandi possono estendersi su due indirizzi allineati e richiederebbero quindi più di un recupero sul bus dati. È possibile che le CPU supportino questo, ma questa funzionalità è raramente richiesta direttamente a livello di codice macchina, quindi i progettisti di CPU normalmente evitano di implementarla e invece emettono errori di bus per l'accesso alla memoria non allineata.

- errore di paging
  FreeBSD, Linux e Solaris possono segnalare un errore del bus quando le pagine di memoria virtuale non possono essere impaginate, ad esempio perché è scomparso (ad esempio l'accesso a un file mappato alla memoria o l'esecuzione di un'immagine binaria che è stata troncata mentre il programma era in esecuzione), o perché un file mappato in memoria appena creato non può essere allocato fisicamente, perché il disco è pieno.

- segmento non presente
  Su x86 esiste un vecchio meccanismo di gestione della memoria noto come segmentazione. Se l'applicazione carica un registro di segmento con il selettore di un segmento non presente , l'eccezione viene generato.