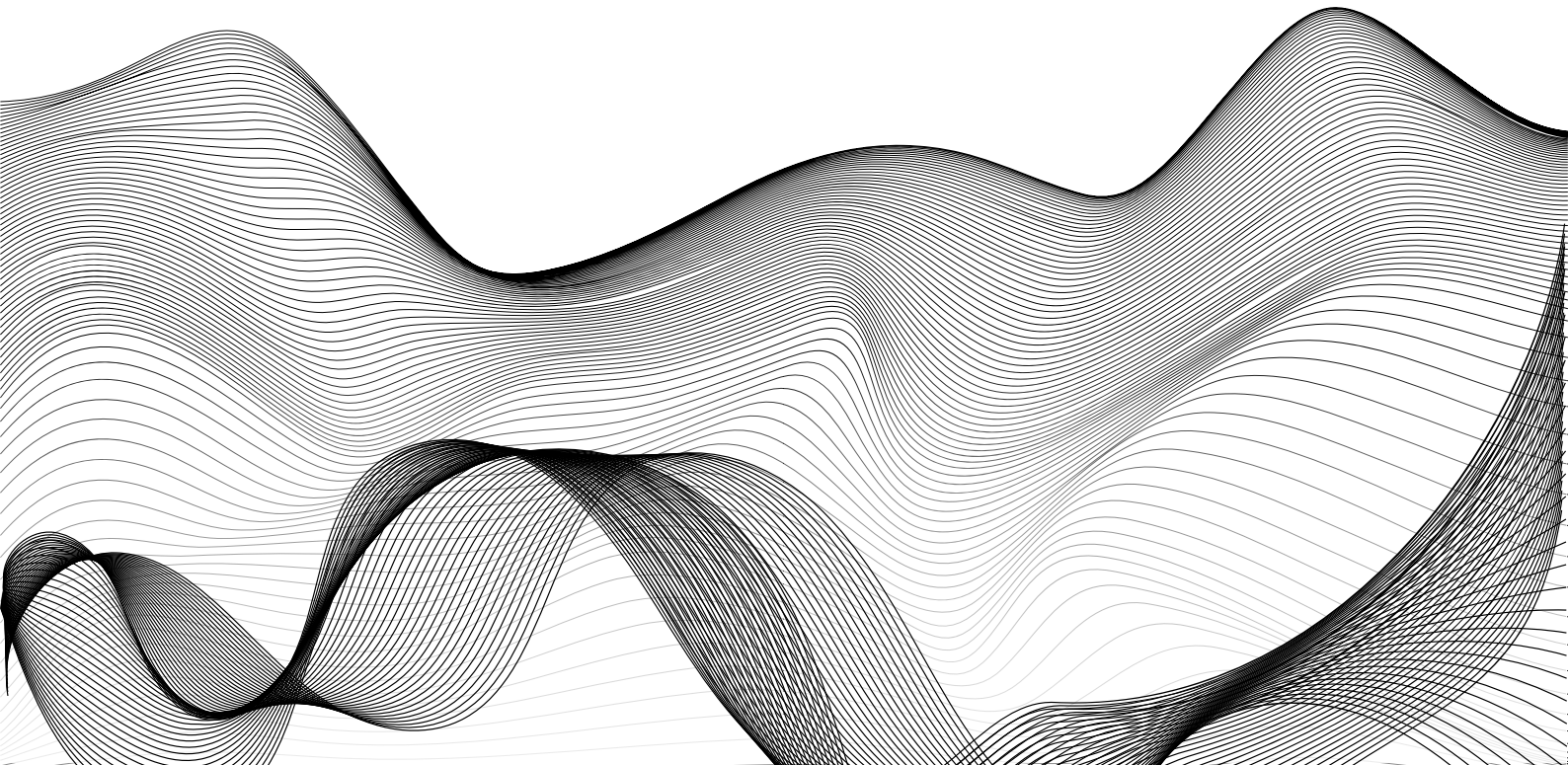
ELEONORA
VIOLA

DATASHIELDS

BUFFER OVERFLOW

S7/L4



TRACCIA

Abbiamo già parlato del 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).

SEGMENTATION FAULT

Un segmentation fault (errore di segmentazione) è un tipo di errore di runtime che si verifica quando un programma tenta di accedere a una porzione di memoria non consentita.

Questo può accadere per diversi motivi, ma i più comuni includono:

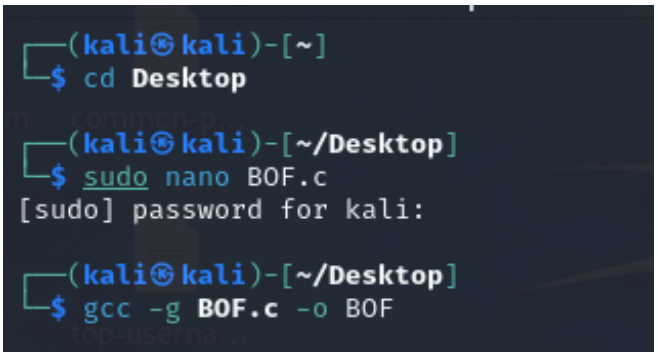
1. **Accesso a memoria non allocata:** tentare di leggere o scrivere in una parte di memoria che non è stata allocata dal programma.
2. **Buffer Overflow:** scrivere più dati di quanto il buffer (una porzione di memoria) possa contenere, causando la sovrascrittura di altre aree di memoria.
3. **Accesso a puntatori nulli:** utilizzare puntatori che non sono stati inizializzati o che sono stati impostati a NULL.
4. **Accesso a memoria libera:** tentare di utilizzare memoria che è stata già deallocata con free o delete.

Quando si verifica un segmentation fault, il sistema operativo interrompe l'esecuzione del programma per prevenire ulteriori danni o comportamenti imprevedibili.

SVOLGIMENTO

Nelle slide possiamo trovare lo script con cui dobbiamo interagire:

Possiamo procedere con il creare un documento (BOF.c) e scrivere il codice.
Per compilare il file si usa il comando gcc **–g BOF.c –o BOF**



SVOLGIMENTO

Adesso possiamo eseguire il programma eseguendo il comando ./BOF.

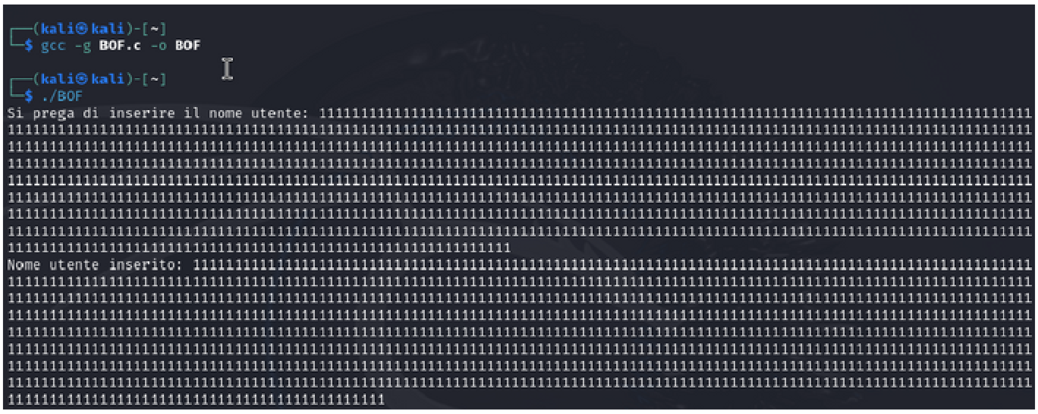
Possiamo notare che inserendo meno di 10 caratteri, non si presenta l'errore di segmentazione.

Continuando a fare prove e inserendo più di 10 caratteri il codice ha un comportamento insolito e non riporta il **segmentation fault:**

L'esempio sopra presenta 20 caratteri.

SVOLGIMENTO

Qui un altro esempio con ben 999 caratteri e ancora nessun errore:



L'assenza di un errore di segmentazione in questo caso, anche quando si supera la dimensione del buffer, potrebbe essere dovuta a una serie di fattori legati al comportamento indefinito del buffer overflow e alla gestione della memoria del sistema operativo e del compilatore.

Come ad esempio:

- **Allocazione della memoria:** Gli ambienti di runtime moderni, come quelli su macOS, possono allocare blocchi di memoria più grandi di quelli richiesti dal programma per migliorare le prestazioni e la gestione della memoria. Questo significa che anche se si scrive oltre il limite del buffer, si potrebbe non sovrascrivere immediatamente una memoria critica, evitando un errore di segmentazione immediato.

- **Protezione della memoria**: Alcuni sistemi operativi utilizzano tecniche di protezione della memoria che possono ritardare o mitigare il verificarsi di errori di segmentazione.

- **Comportamento indefinito:** Il comportamento di un programma che causa un buffer overflow è indefinito secondo lo standard del linguaggio C. Questo significa che il programma potrebbe comportarsi in modo apparentemente corretto o in modo completamente inaspettato senza causare un errore di segmentazione immediato.

- **Caratteristiche Specifiche dell'Emulatore UTM:** dato che sto eseguendo il programma all'interno di un ambiente virtualizzato UTM su un Mac M1, l'emulazione potrebbe influenzare il modo in cui vengono gestiti gli errori di memoria, rendendo meno probabile che un errore di segmentazione venga rilevato.

CONCLUSIONE

Per migliorare lo script si potrebbe modificare il codice per utilizzare una funzione di input sicura come **fgets**, che permette di specificare la dimensione del buffer, evitando così l'overflow:

```c
#include <stdio.h>

int main() {
    char buffer[10];

    printf("Si prega di inserire il nome utente: ");
    fgets(buffer, sizeof(buffer), stdin);

    printf("Nome utente inserito: %s\n", buffer);

    return 0;
}
```

Infatti come si può vedere dall'esempio in basso il codice blocca efficacemente.