Cybersecurity: Buffer Overflow

Lezione 1: Introduzione e Memory Layout

Introduzione

- Cos'è un buffer overflow
- Perché è pericoloso
- Esempio storico: DilDog's Tao (1998)

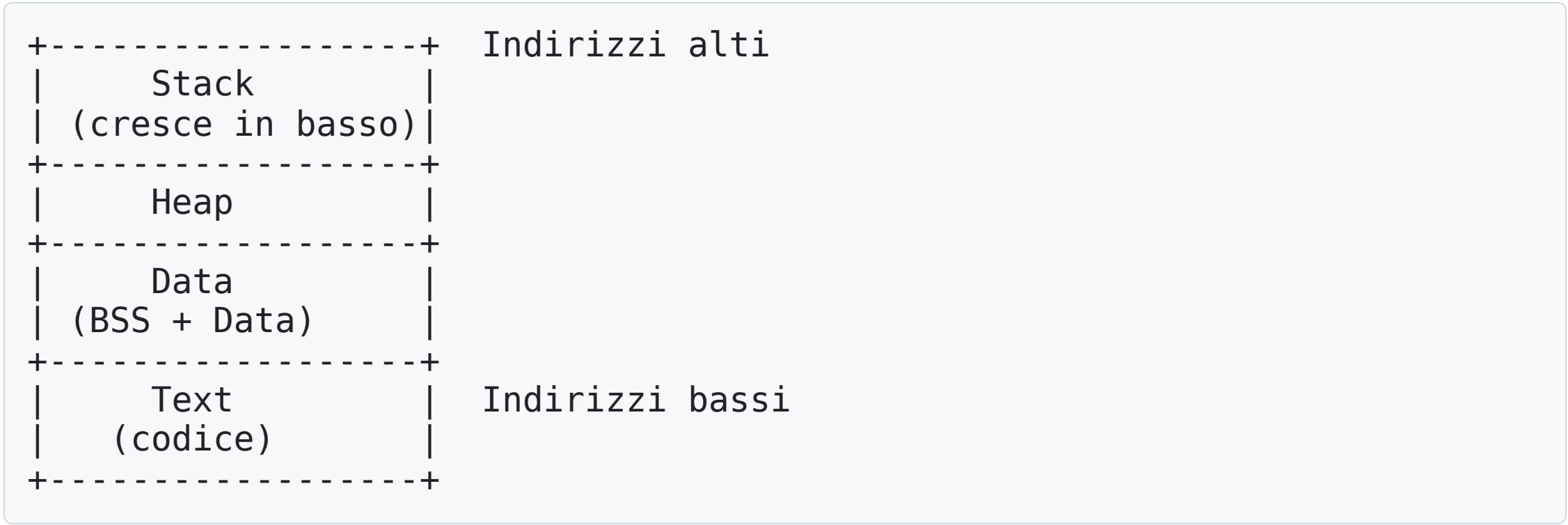
Buffer Overflow

Un buffer overflow si verifica quando un programma scrive più dati in un buffer di quanto questo possa contenere, sovrascrivendo la memoria adiacente.

Possibili conseguenze:

- Crash del programma
- Corruzione dati
- Esecuzione di codice arbitrario

Memory Layout di un Processo

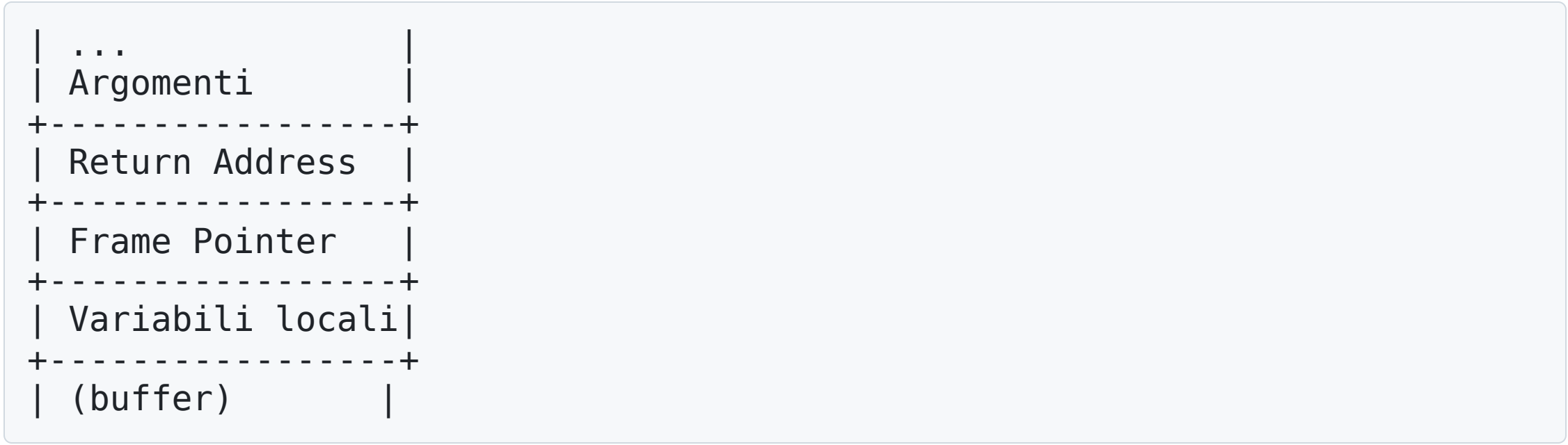


Segmenti di Memoria

- **Text**: codice eseguibile (read-only)
- **Data**: variabili globali/statiche
- **Heap**: allocazione dinamica (malloc, new)
- **Stack**: variabili locali, chiamate di funzione

Stack Frame

Ogni funzione crea uno stack frame:



Esempio C Vulnerabile

```c
#include <stdio.h>
#include <string.h>

void vulnerable() {
    char buffer[64];
    strcpy(buffer, getenv("INPUT"));  // pericoloso!
    printf("Echo: %s\n", buffer);
}
```

Cosa Succede

Se INPUT > 64 byte:

- strcpy continua a scrivere
- Sovrascrive EBP e EIP (return address)
- Controllo del flusso di esecuzione

The Tao of Buffer Overflow

Articolo di DilDog (cDc, 1998):

- Classico sullo sfruttamento Windows
- Shellcode di 112 byte con XOR
- Tecniche ancora attuali

Obiettivi dello Sfruttamento

1. Trovare offset per raggiungere RET
2. Iniettare shellcode
3. Reindirizzare EIP a nostro codice

NOP Sled

```
[ NOP (0x90) ... NOP ][shellcode][indirizzo]
          ^
          +-- salto "scivola" su NOP
```

Summary

- Buffer overflow = overflow di buffer → corruzione memoria
- Stack layout: buffer | EBP | RET
- Obiettivo: controllare RET
- Tecnica: NOP sled + shellcode

Esercizio

Compilare:

```
gcc -m32 -fno-stack-protector -z execstack -no-pie -o vuln vuln.c
```

Usare GDB per trovare l'offset e testare overflow.

Riferimenti

- DilDog, *The Tao of Buffer Overflow* (1998)
- Aleph One, *Smashing The Stack For Fun And Profit* (1996)

Cybersecurity: Buffer Overflow – Pinolallo.com