BUFFER OVERFLOW

Un buffer è una regione di memoria contigua che archivia dati dello stesso tipo.

Un buffer overflow avviene quando i dati sono scritti oltre la fine del buffer.

Il danno risultante dipende da:

- Dove i dati vanno a finire.
- Come è usata questa regione di memoria.
- che modifiche sono fatte.

LAYOUT DELLA MEMORIA VIRTUALE

Lo Stack cresce verso il basso e contiene parametri di funzione, variabili locali, vari indirizzi.

L'heap cresce verso l'alto e contiene dati allocati dinamicamente.

DIFESE

- USARE PROGRAMMAZIONE DIFENSIVA: 1. evitare funzioni di libreria pericolose e sostituirle con varianti sicure (gets → fgets, strcpy → strncpy, puts → fputs, sprintf → snprintf)
2. Controllare sempre il limite degli array quando ci itera su di essi.

INSERIRE UN CANARY

Un canary è un valore nello stack il cui valore è testato prima di usare return.

È un valore random/mio e un valore composto da alcuni terminatori di stringa (deve essere difficile da indovinare per l'attaccante)

ex.

```
void function (...) {
    int canary;
    char buff [MAX_SIZE];
    canary = CANARY_VALUE;
    gets (buff);
    ...
    if (canary != CANARY_VALUE) exit
    return; }
```

EVITA C++: Usa un linguaggio type safe dove la lunghezza di un array fa parte del suo tipo e assegnare il contenuto di un buffer ad uno più piccolo è un errore di tipo. (JAVA)

- EVITA I BUFFER SULLO STACK: Invece uso lo spazio nell'heap.

ESERCIZI

ESERCIZIO 5 BUFFER OVERFLOW, ES. ESAME

bisogno usare strncpy al posto di strcpy perché è SAFE

ESERCIZIO 5 BUFFER OVERFLOW, 14/01/2019

A) write a program (in c) suffering from a buffer overflow

```
main() {
    char buff[5];
    sprintf(buff, "abcdef");
    puts(buff);
} return 0;
```

B) Modify the program so to prevent the buffer overflow

```
main() {
    char buff[5];
```

```
    snprintf (buff, sizeof(buff), "abcdef");
    fputs(buff);
}   return 0;
```