

S7L4

Sarah Ortiz



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)

Seconda parte della traccia:

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

- Aumentando la dimensione del vettore a 30;

```
(kali@kali)-[~/Desktop]
$ sudo nano bof.c

(kali@kali)-[~/Desktop]
$ gcc -g bof.c -o bof

(kali@kali)-[~/Desktop]
```

Per prima cosa va creato un file tramite il comando:

“sudo nano bof.c”,
dopo di che con il comando
“gcc -g bof.c -o bof”

facciamo invocare il compilatore GNU per il linguaggio C e includiamo informazioni di debug nel file oggetto generato.

```
File Actions Edit View Help
GNU nano 8.0 bof
#include <stdio.h>

int main () {

char buffer [10];

printf ("si prega di inserire il nome utente:");
scanf ("%s", buffer);
printf ("nome utente inserito:%s\n", buffer);

return 0;
}
```

(kali@kali)-[~/Desktop]

\$./bof

si prega di inserire il nome utente:sarahortiz
nome utente inserito:sarahortiz

(kali@kali)-[~/Desktop]

\$./bof

si prega di inserire il nome utente:sarahortiz24042001
nome utente inserito:sarahortiz24042001
zsh: segmentation fault ./bof

(kali@kali)-[~/Desktop]

\$

```
(kali@kali)-[~/Desktop]
```

```
$ ./bof
```

```
si prega di inserire il nome utente:sarahortiz
```

```
nome utente inserito:sarahortiz
```

```
(kali@kali)-[~/Desktop]
```

```
$ ./bof
```

```
si prega di inserire il nome utente:sarahortiz24042001
```

```
nome utente inserito:sarahortiz24042001
```

```
zsh: segmentation fault ./bof
```

```
(kali@kali)-[~/Desktop]
```

```
$ ./bof.c
```

Possiamo notare come dipendendo da byte utilizzati possiamo causare un buffer oppure no. Nel secondo esempio l'input supera i 10 byte del buffer, causando un overflow e, potenzialmente, scrivendo su aree di memoria non consentite.

File Actions Edit View Help

GNU nano 8.0

bof30.c

```
//SECONDA PARTE DOVE AUMENTO LA DIM DEL BUFFER A 30 BYTE
#include <stdio.h>
#include <string.h>

void vulnerable_function () {

char buffer [30];

printf ("si prega di inserire il nome:");
gets (buffer);
printf ("nome inserito:%s\n", buffer);

}

int main () {
vulnerable_function();
printf ("fine programma");

return 0;
}
```

Per la seconda parte della traccia richiesta, creiamo un nuovo file con il codice che aumenta la dimensione del buffer a 30 byte

```
(kali㉿kali)-[~/Desktop]
$ gcc -g bof30.c -o bof30
bof30.c: In function 'vulnerable_function':
bof30.c:10:1: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
  10 | gets (buffer);
      | ^~~~
      | fgets
/usr/bin/ld: /tmp/ccQqJ1jm.o: in function `vulnerable_function':
/home/kali/Desktop/bof30.c:10:(.text+0x29): warning: the `gets' function is dangerous and should not be used.
```

```
(kali㉿kali)-[~/Desktop]
$ ./bof30
si prega di inserire il nome:ciaomichiamosarahortiz
nome inserito:ciaomichiamosarahortiz
fine programma
```

```
(kali㉿kali)-[~/Desktop]
$ ./bof
si prega di inserire il nome utente:ciaomichiamosarahortiz
nome utente inserito:ciaomichiamosarahortiz
zsh: segmentation fault ./bof
```

Ho messo a confronto
entrambi i file con i codici
di buffer diversi e
possiamo notare come il
prima con il buffer a 30
byte non presenta errori