

Brève introduction

Tout d'abord qu'est ce qu'un buffer overflow? Et à quoi ça sert?

Les buffer overflow sont des failles nous permettant d'exploiter une vulnérabilité présente dans un fichier binaire afin de pouvoir lui faire exécuter ce que l'on veut (enfin quasiment ^^).

Le plus souvent on lui fait exécuter un shell, chose qui peut être très utile, par exemple si le binaire a des permissions SUID ou si il tourne sur un port en remote.

Quelques notions de base à connaître avant d'attaquer cet article:

- La gestion de la mémoire
- Le fonctionnement de la stack
- Et quelques bases en assembleur

Rappel

Pour commencer je vais vous faire un petit rappel sur le déroulement de notre pile (stack) sur un programme qui va tout simplement attendre une entrée utilisateur (stdin) pour ensuite nous l'afficher (stdout)

```
#include <unistd.h>
#include <stdio.h>

void vuln() {
    char buffer[32];
    read(0, buffer, 128);
    puts(buffer);
}

int main() {
    vuln();
}
```

Nous allons ensuite compiler ce programme sans aucune protections, et en 32bit

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

Executons ce programme:

```
$ ./vuln
AAAAAAAAAAAA
AAAAAAAAAAAA
$
```

Tout se déroule comme prévu

Et voici à quoi la stack ressemblait lorsque nous avons entrée nos "A":

```
0xbfff0000: 41 41 41 41
0xbfff0004: 41 41 41 41
0xbfff0008: 41 41 41 00
...
0xbfff0020: 30 00 ff bf <- sEBP
0xbfff0024: f0 84 04 08 <- sEIP
```

Si pour vous tout est bon on peut donc passer à la suite!

Revision #6

Created 6 May 2020 13:09:26 by z3pp

Updated 7 May 2020 15:01:07 by z3pp