Return Oriented Programming

LYS

2020-03-31

ROP

- 當 binary 開啟 NX 保護時，無法執行 shellcode，就會需要 ROP
  - 基本上現在預設都會有 NX
- 通常用來開 shell 或者拿到一塊 RWX 的記憶體來跑 shellcode
  - Shellcode 還是比 ROP 好用
  - 通常用 mprotect 或 mmap 來拿 RWX

ROP gadget

- ROP 就像玩拼圖，把一塊一塊的 code 片段組合起來
- ROP gadget 就是各種 code 片段
- 可以從 binary 中找 C3 CB C2 CA 這幾個 byte，找到這些 byte，往前取幾個 byte，做 disassembly，看看有沒有好用的 gadget
- ROPgadget tool
  - https://github.com/JonathanSalwan/ROPgadget
  - 現成的工具幫你找 gadget
  - $ ROPgadget --binary ./binary

ROP gadget

```
0x000000000040213a : pop rdi ; pop rbp ; ret
0x0000000000401516 : pop rdi ; ret
0x0000000000446baa : pop rdx ; add byte ptr [rax - 0x7f], cl ; ret 0x5a40
0x00000000004a3429 : pop rdx ; clc ; jmp qword ptr [rax]
0x00000000004ba341 : pop rdx ; je 0x4ba30b ; ret 0xacfc
0x00000000004ba453 : pop rdx ; leave ; retf
0x00000000004bb307 : pop rdx ; out 0xf5, eax ; mov ch, 0xab ; ret
0x00000000004428e4 : pop rdx ; pop r10 ; ret
0x0000000000478617 : pop rdx ; pop rbx ; ret
0x0000000000446ba9 : pop rdx ; pop rdx ; add byte ptr [rax - 0x7f], cl ; ret 0x5a40
0x0000000000442909 : pop rdx ; pop rsi ; ret
0x00000000004428e6 : pop rdx ; ret
0x0000000000414202 : pop rdx ; retf
0x00000000004bae70 : pop rdx ; sub ah, ah ; mov bh, 6 ; movsd dword ptr [rdi], dword ptr [rsi] ; int1 ; ret
0x5803
0x000000000042ead4 : pop rsi ; adc byte ptr [rsi + 0xf], ah ; jmp 0x42eaaa
0x000000000042eb52 : pop rsi ; adc byte ptr [rsi + 0xf], ah ; jmp 0x42eb28
0x000000000042efe4 : pop rsi ; adc byte ptr [rsi + 0xf], ah ; jmp 0x42efba
0x000000000042f062 : pop rsi ; adc byte ptr [rsi + 0xf], ah ; jmp 0x42f038
0x000000000042be6e : pop rsi ; add al, 0 ; add byte ptr [rax - 0x7d], cl ; ret 0xe910
0x00000000004739d3 : pop rsi ; add dword ptr [rax], eax ; add byte ptr [rax + 0x39], cl ; ret 0xf48
0x00000000004261ec : pop rsi ; add dword ptr [rax], eax ; ret
```

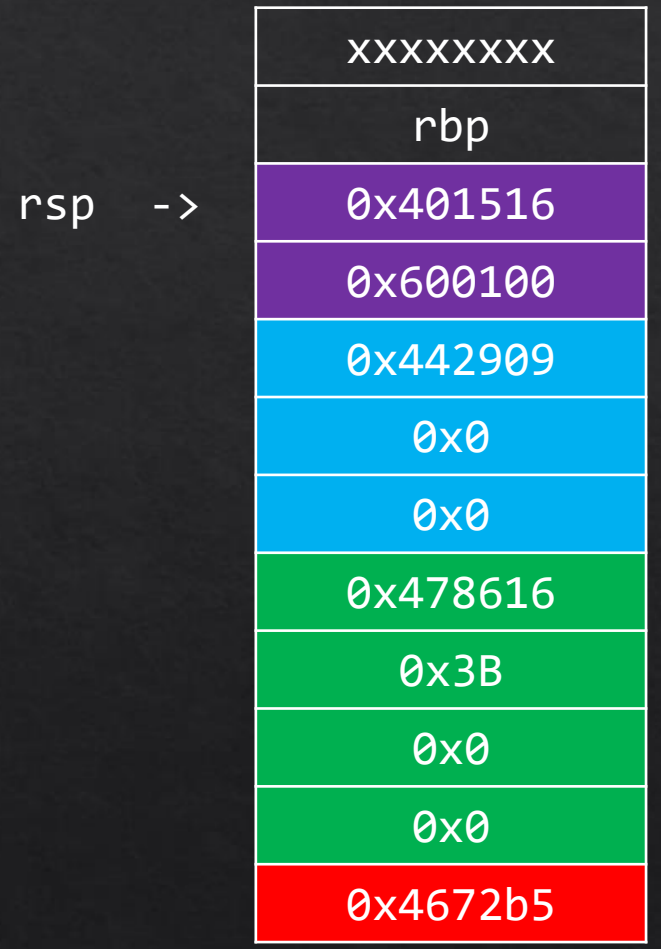
ROP Chain

- execve(“/bin/sh”,NULL,NULL)
- register
  - rax = 59
  - rdi = address of “/bin/sh”
  - rsi = NULL
  - rdx = NULL
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	xxx
rsi	xxx
rdx	xxx
rip	xxx



ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	xxx
rsi	xxx
rdx	xxx
rip	0x401516

rsp ->

xxxxxxxx
rbp
0x401516
0x600100
0x442909
0x0
0x0
0x478616
0x3B
0x0
0x0
0x4672b5

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	0x600100
rsi	xxx
rdx	xxx
rip	0x401516

rsp ->

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	0x600100
rsi	xxx
rdx	xxx
rip	0x442909

rsp ->

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	0x600100
rsi	xxx
rdx	0x0
rip	0x442909

rsp ->

xxxxxxxx
rbp
0x401516
0x600100
0x442909
0x0
0x0
0x478616
0x3B
0x0
0x0
0x4672b5

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	0x600100
rsi	0x0
rdx	0x0
rip	0x442909

rsp ->

xxxxxxxx
rbp
0x401516
0x600100
0x442909
0x0
0x0
0x478616
0x3B
0x0
0x0
0x4672b5

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	xxx
rdi	0x600100
rsi	0x0
rdx	0x0
rip	0x478616

rsp ->

xxxxxxxx
rbp
0x401516
0x600100
0x442909
0x0
0x0
0x478616
0x3B
0x0
0x0
0x4672b5

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	0x3B
rdi	0x600100
rsi	0x0
rdx	0x0
rip	0x478616

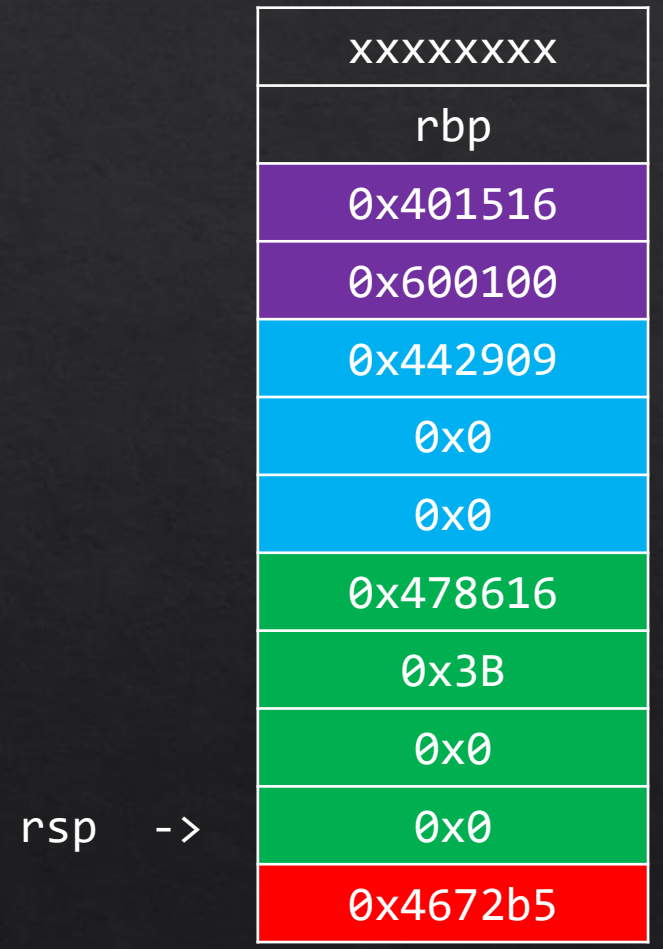
rsp ->

xxxxxxxx
rbp
0x401516
0x600100
0x442909
0x0
0x0
0x478616
0x3B
0x0
0x0
0x4672b5

ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

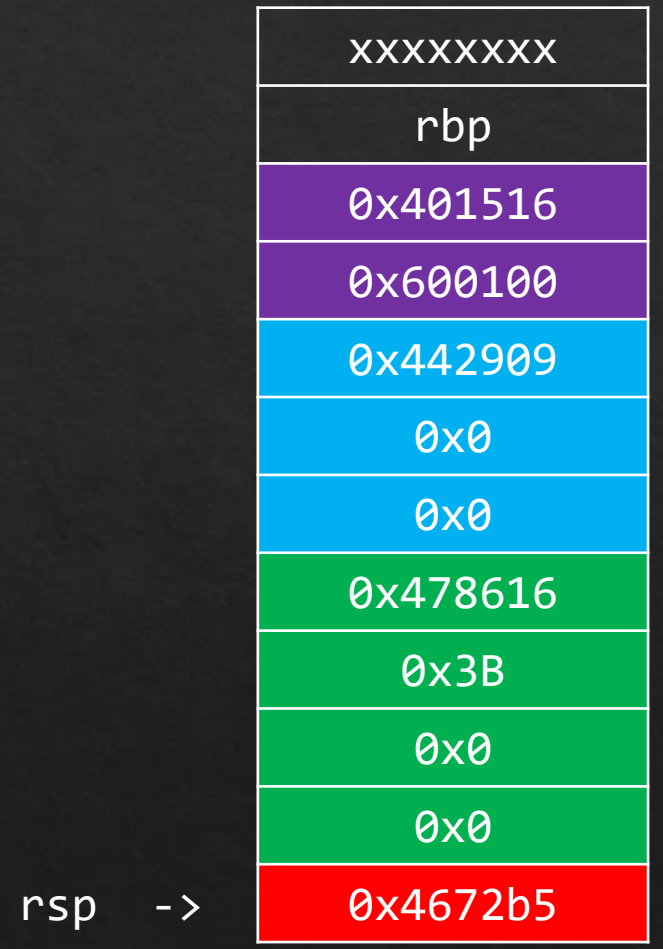
rax	0x3B
rdi	0x600100
rsi	0x0
rdx	0x0
rip	0x478616



ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop rbx ; ret
- 4672b5 : syscall ; ret

rax	0x3B
rdi	0x600100
rsi	0x0
rdx	0x0
rip	0x478616

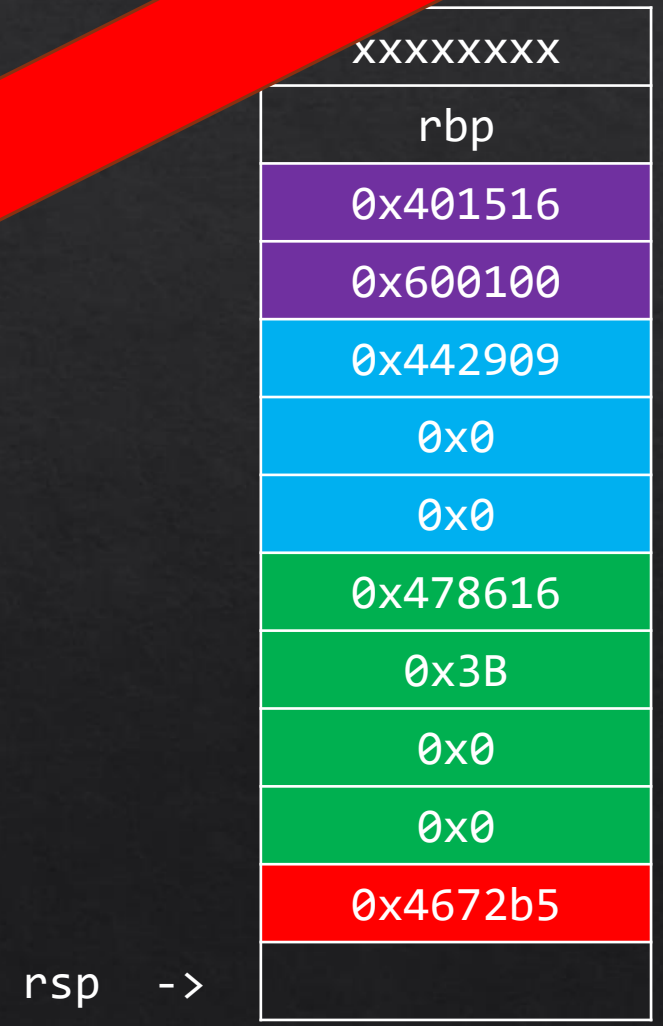


ROP Chain

- ret
- 401516 : pop rdi ; ret
- 442909 : pop rdx ; pop rsi ; ret
- 478616 : pop rax ; pop rdx ; pop
- 4672b5 : syscall ; ret

Get Shell

	0x3B
rdi	0x600100
rsi	0x0
rdx	0x0
rip	0x4672b5



Lab1

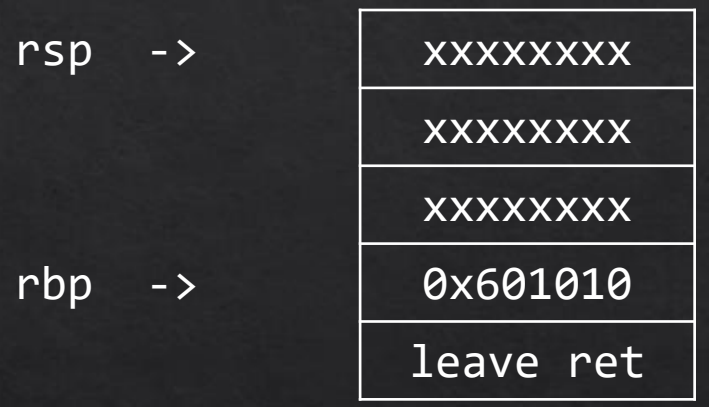
- https://bamboofox.cs.nctu.edu.tw/admin/courses/9/challenges/191

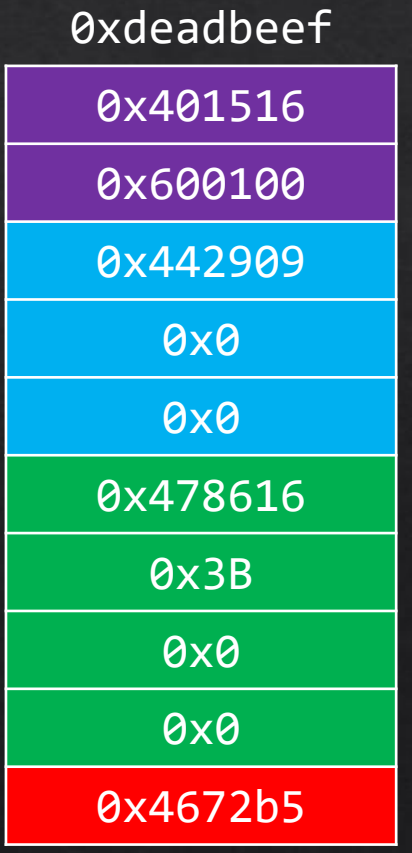
Stack Pivoting

- 也可以說 Stack Migration
- 當 stack 上無法寫入夠長的 gadget 時，可以使用此技巧將 stack 挪到可控的夠長 buffer
- stack 主要是由 rsp 來控制的，所以我們的目標是控制 rsp
- 關鍵的 gadget 是 leave; ret;
  - 這個 gadget 基本上一定找得到
  - leave; ret; 是用來在 call function 結束時恢復上一個 function 的 stack frame
  - leave == mov rsp, rbp; pop rbp;
  - ret == pop rip;

Stack Pivoting

```
leave;
```

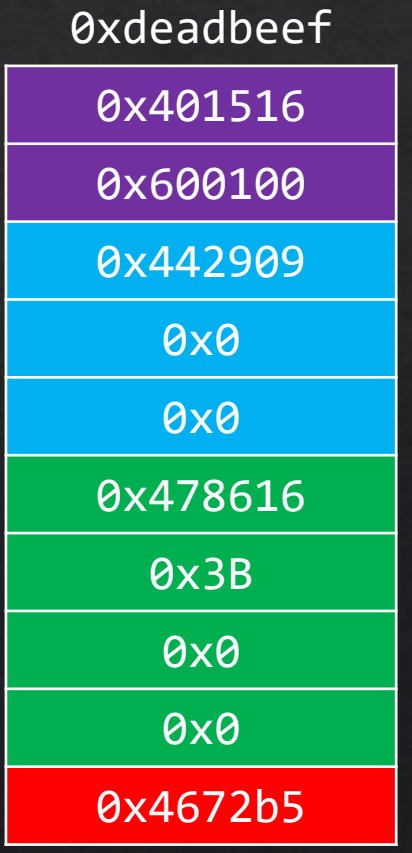




Stack Pivoting

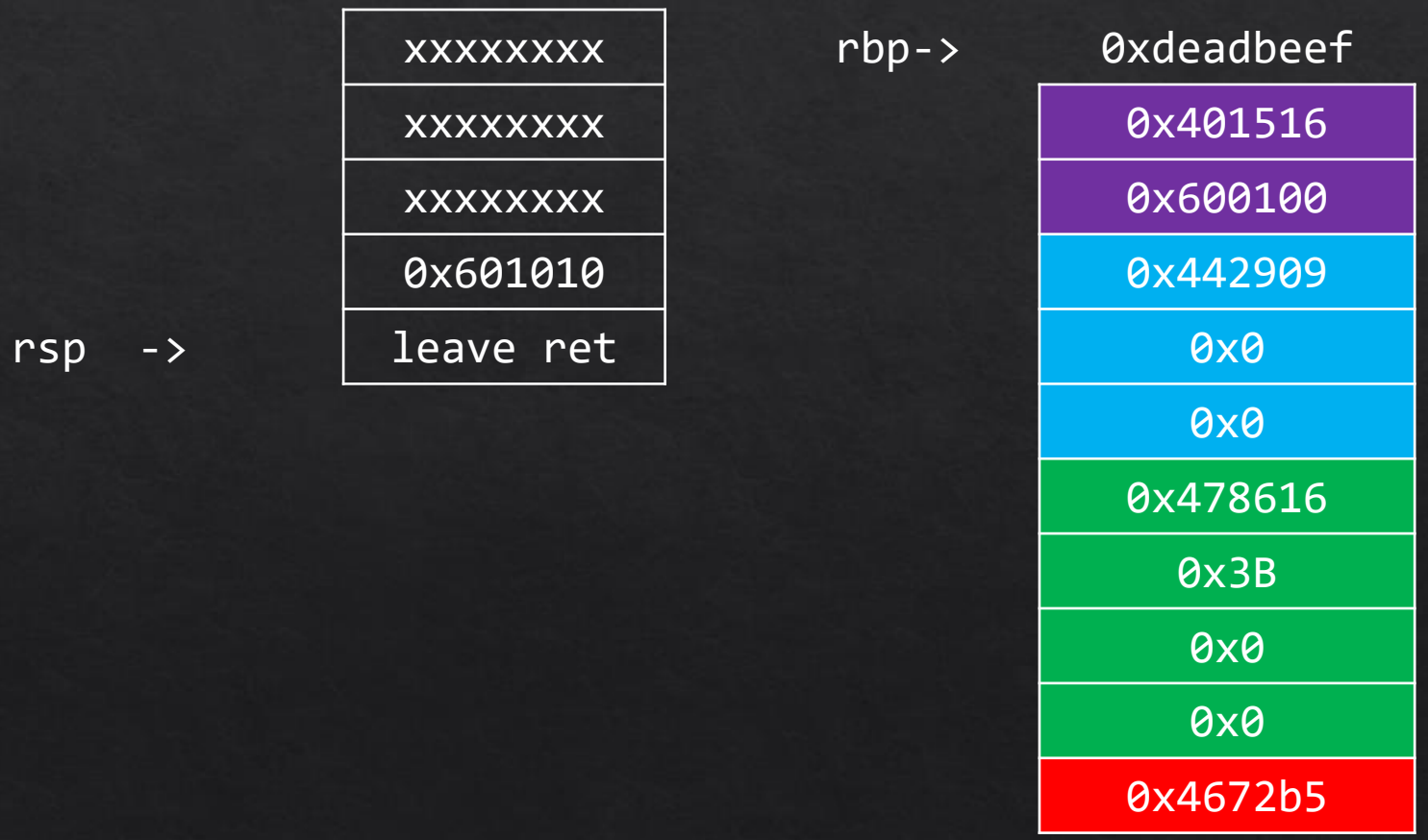
```
leave;
```

rsp,rbp ->



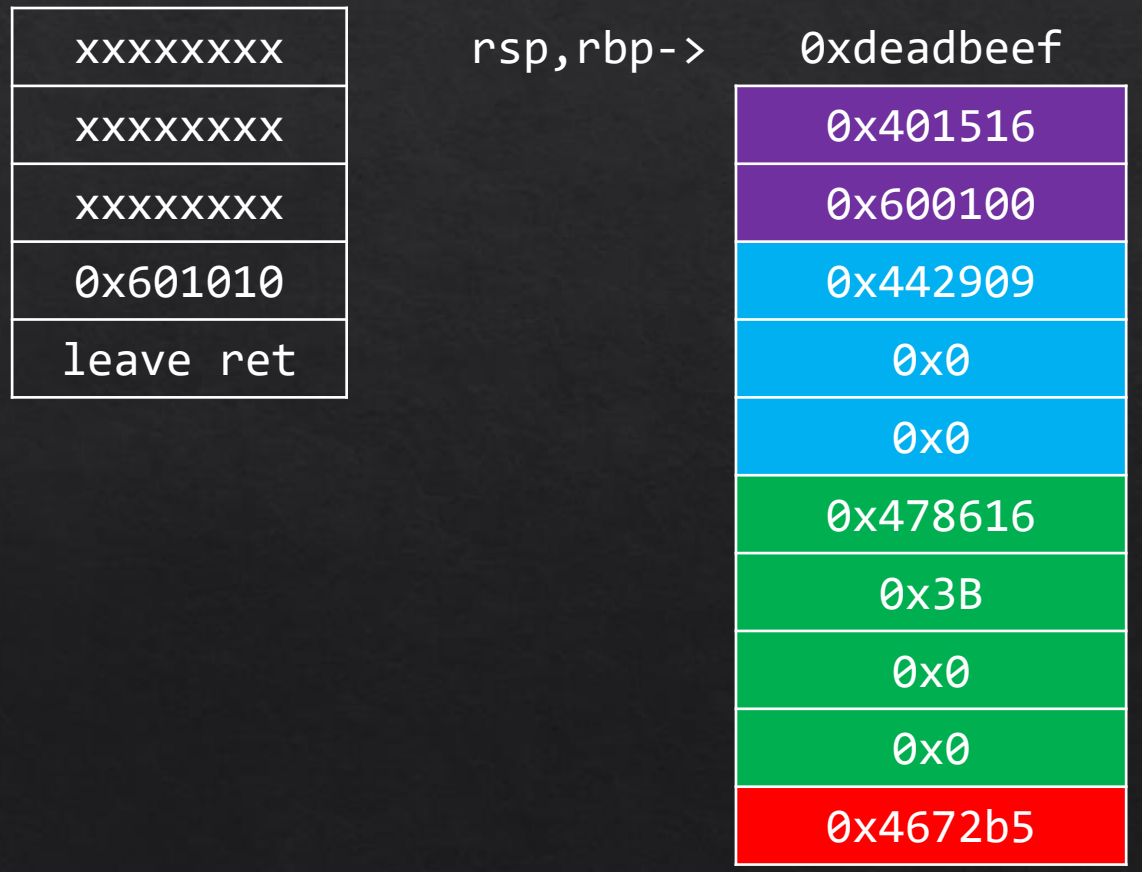
Stack Pivoting

```
ret;
```



Stack Pivoting

```
leave;
```



Stack Pivoting

```
ret;
```

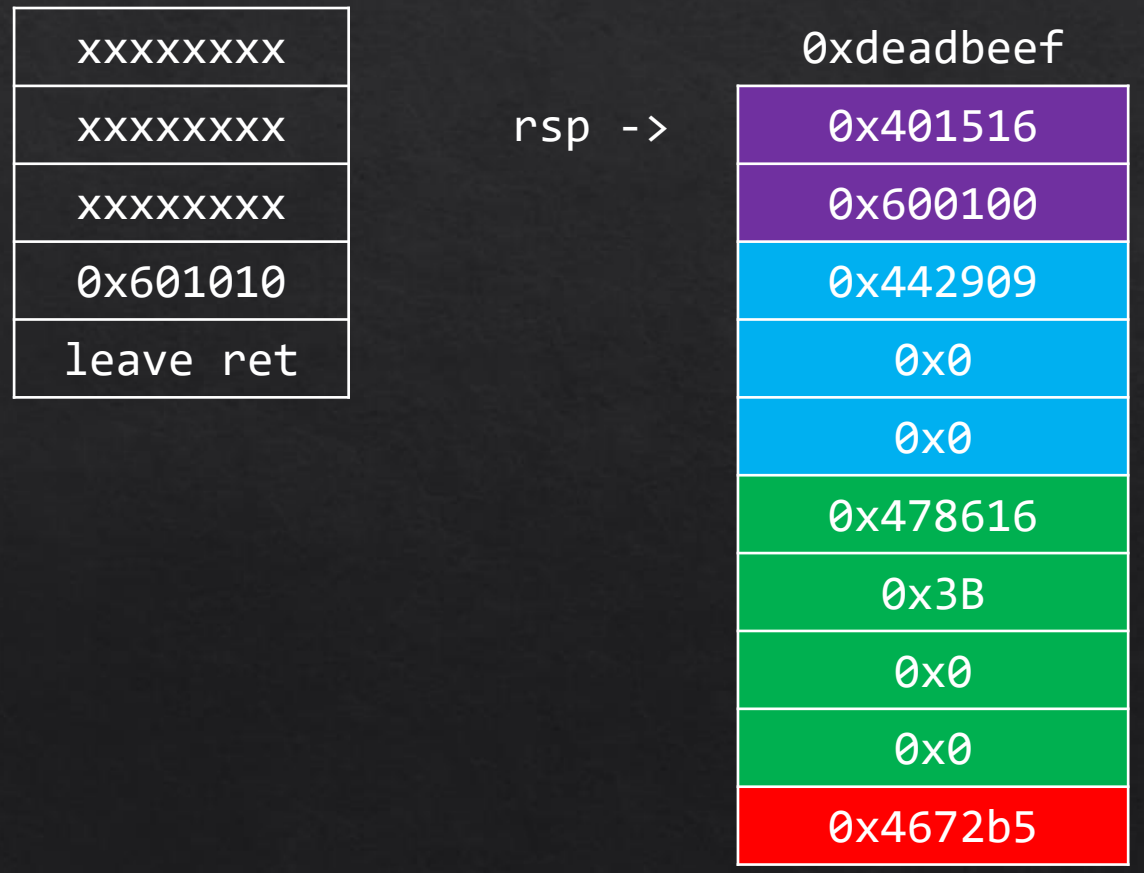
xxxxxxxx
xxxxxxxx
xxxxxxxx
0x601010
leave ret

rsp ->

0xdeadbeef
0x401516
0x600100
0x442909
0x0
0x0
0x478616
0x3B
0x0
0x0
0x4672b5

Stack Pivoting

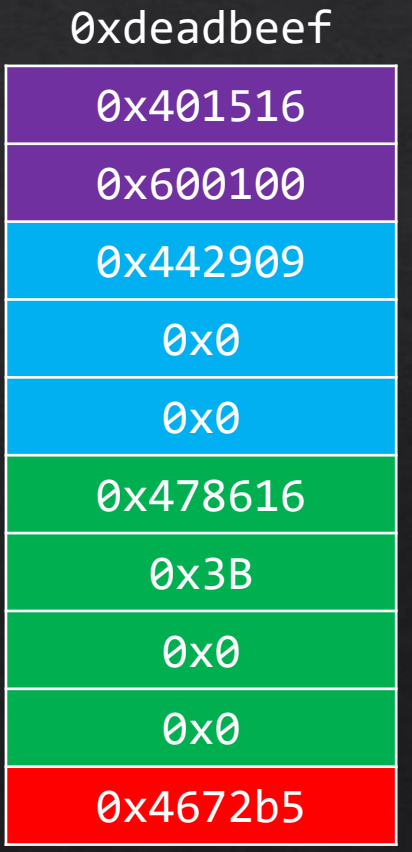
Do rop……



Stack Pivoting

Do rop……

rsp ->



Get Shell

Lab2

- https://bamboofox.cs.nctu.edu.tw/admin/courses/9/challenges/192

Reference

- https://www.felixcloutier.com/x86/ret
- https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/