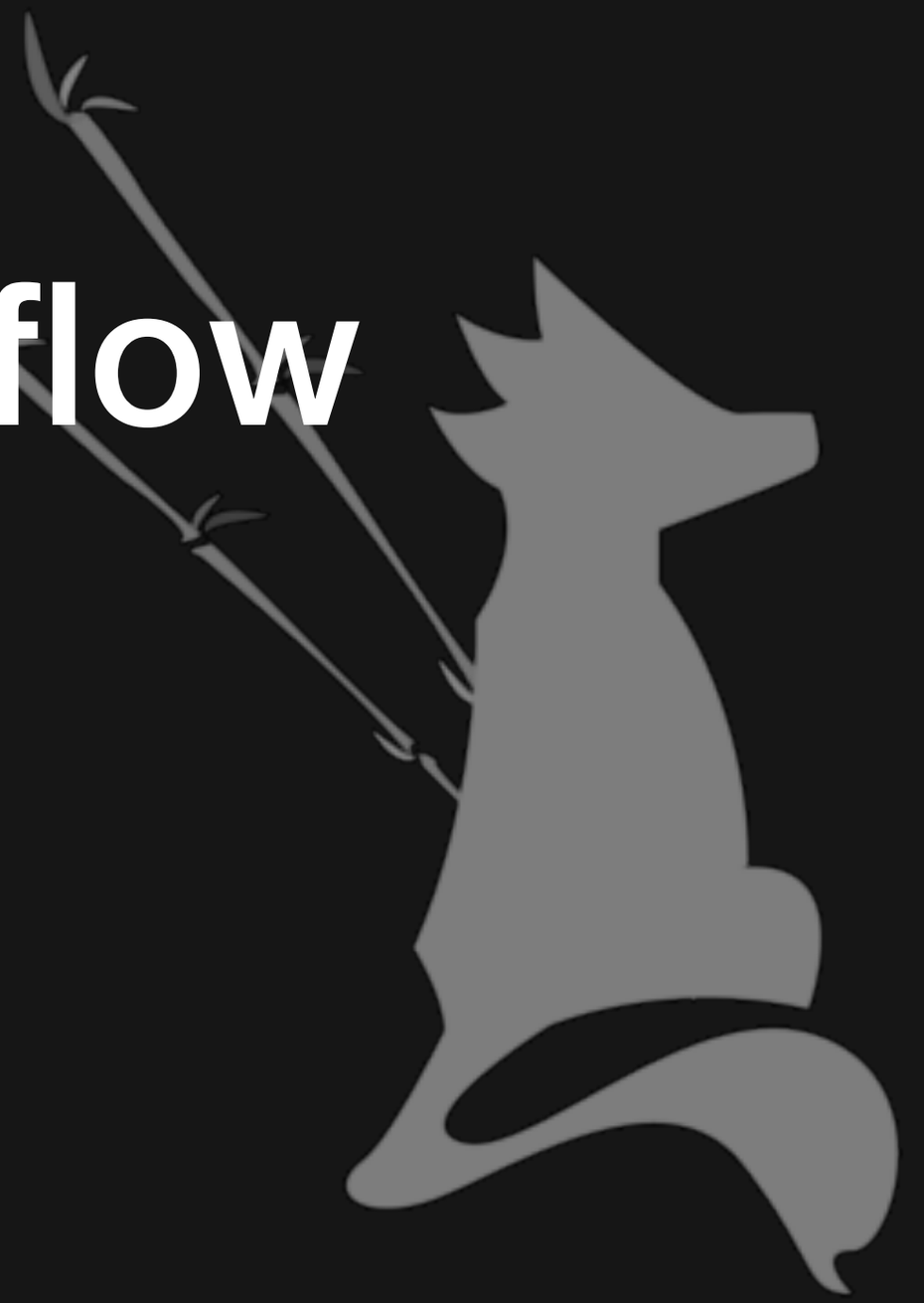
從入門 Pwn 到放棄

frozenkp@BambooFox

Outline

- Buffer Overflow
- Libc
- ROP
- Reverse & Pwn on Go

Buffer Overflow



Buffer Overflow

- 輸入時沒有控制輸入長度，導致記憶體空間被輸入覆蓋掉
- 通常發生在 char 陣列 (字串) 的輸入

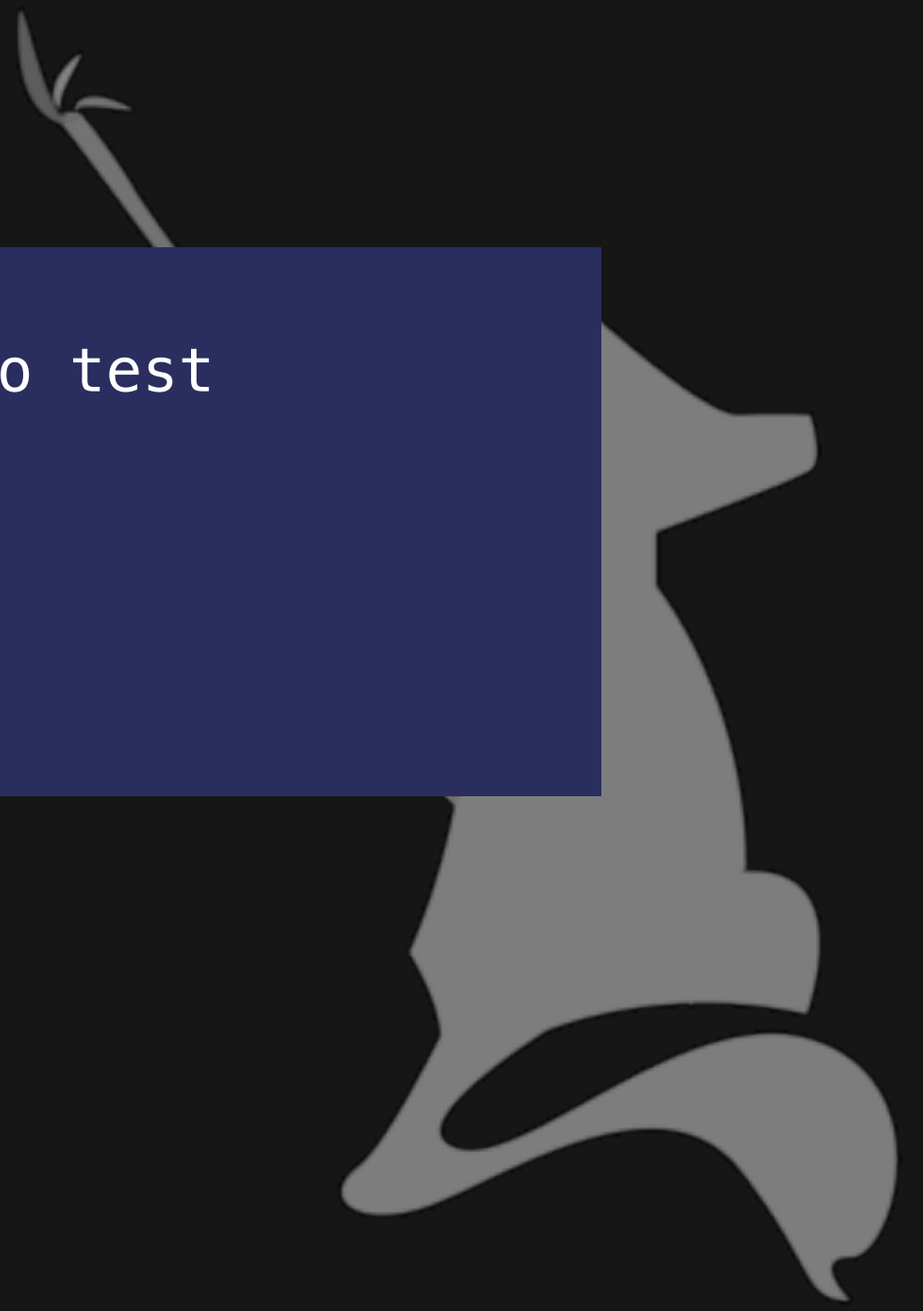
來個🌰

```c
#include <stdio.h>

int main(){
    char buffer[8];
    gets(buffer);          // Input
    puts(buffer);          // Output
    return 0;
}
```

編譯 & 執行

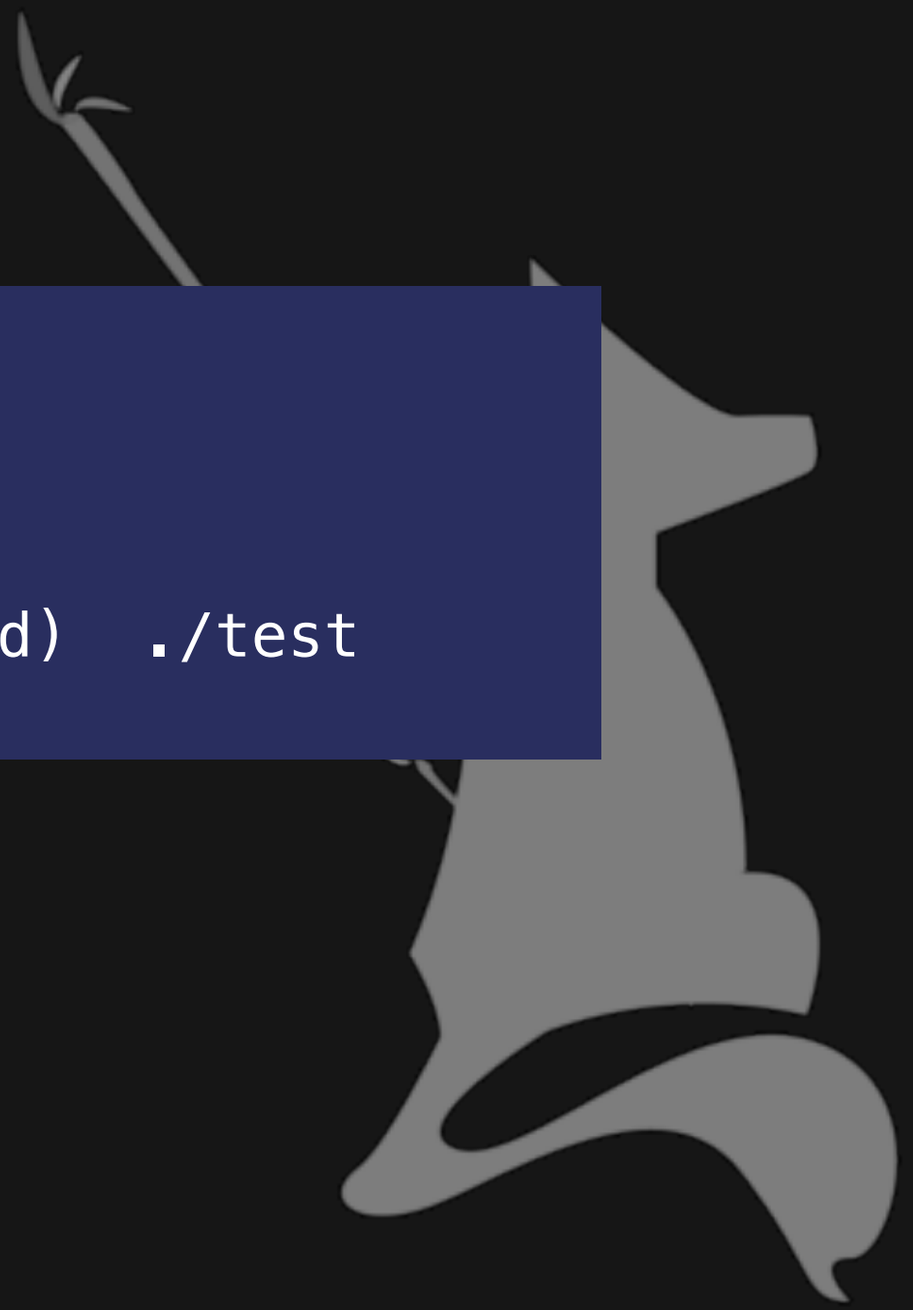
關閉 canary 保護機制

```
% gcc test.c -fno-stack-protector -o test

% ./test
hello
hello
```

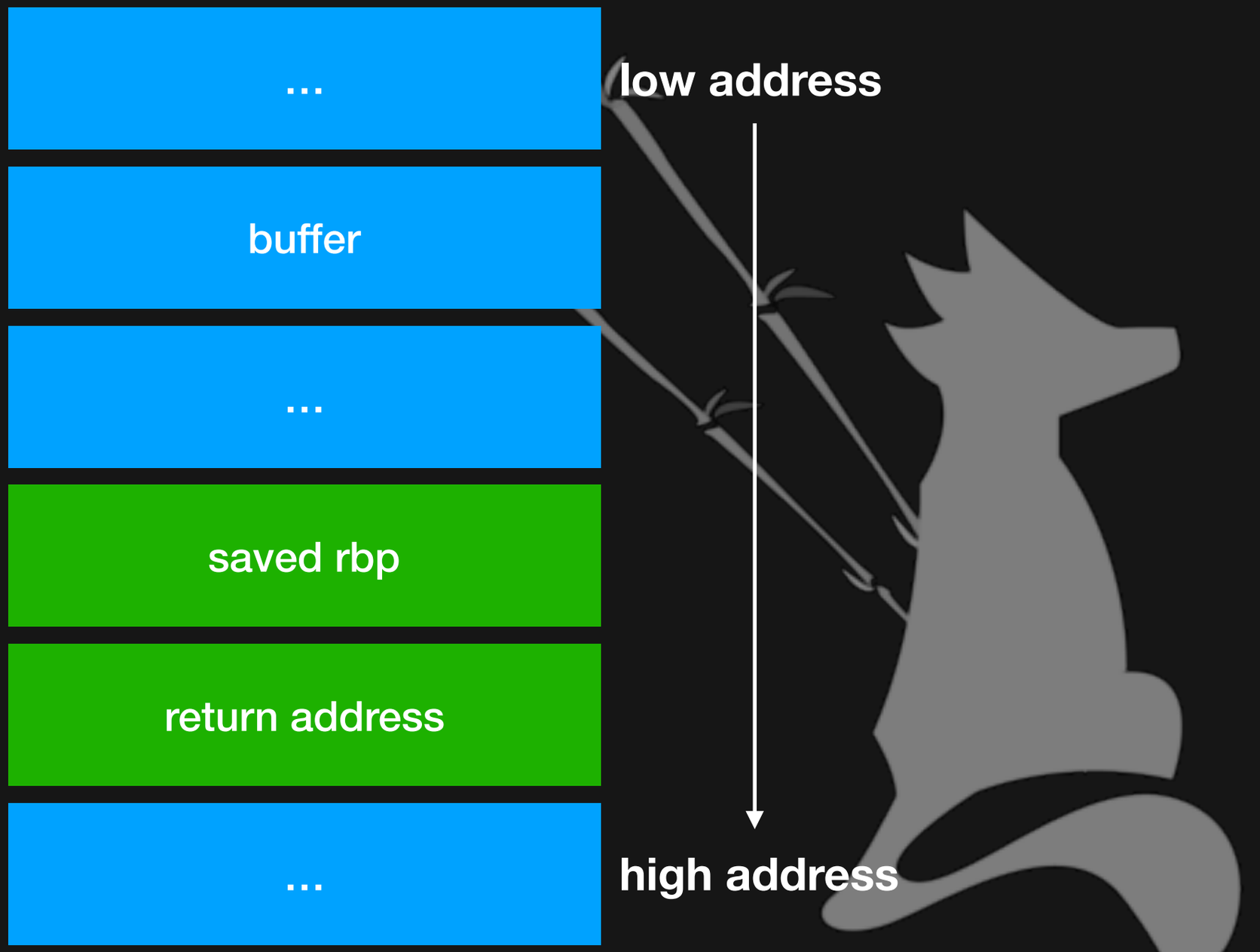


輸入很大的字串？

```
% ./test
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
zsh: segmentation fault (core dumped)  ./test
```



What happened ?



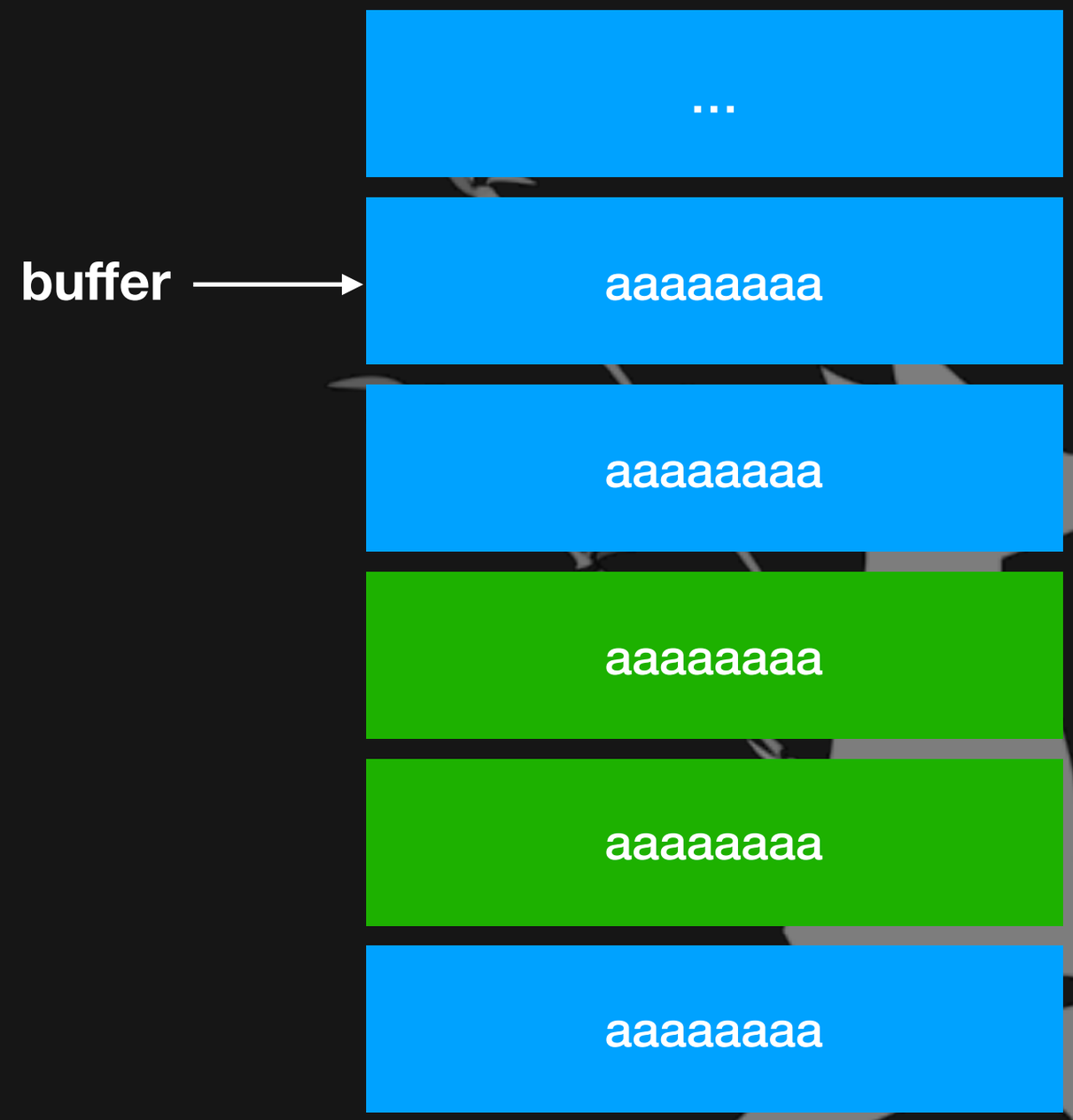
What happened ?

gets & read

- gets
  - 沒有限制輸入長度
- read
  - 有限制最大輸入長度
  - 可 overflow 大小為最大輸入長度與 buffer 長度之間

gets & read

```c
#include <stdio.h>

int main(){
    char buffer[8];
    gets(buffer);
    puts(buffer);
    return 0;
}
```

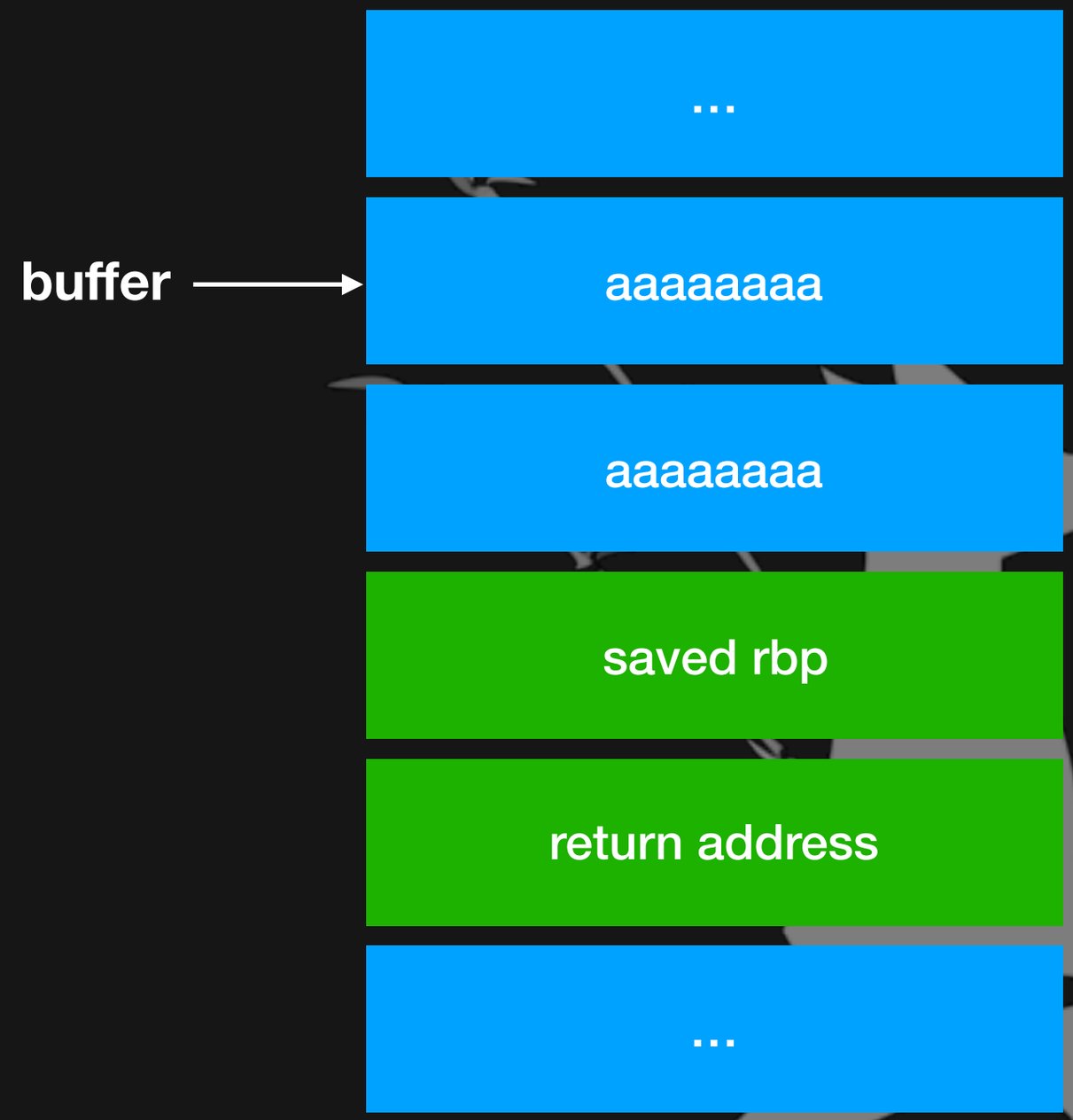


gets & read

```c
#include <stdio.h>

int main(){
    char buffer[8];
    read(0, buffer, 16);
    puts(buffer);
    return 0;
}
```

只能 overflow 8 bytes

最大長度 (16) - buffer 長度 (8) = 8

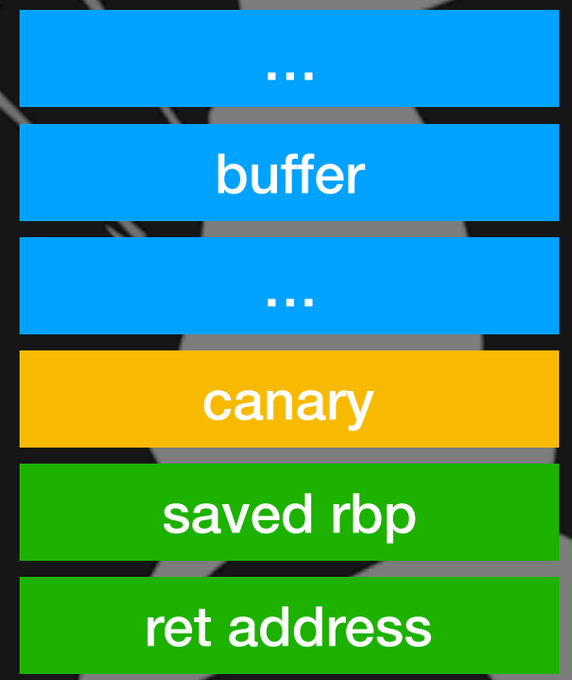


Stack Canary

- 在 rbp 之前塞一個 random 值，ret 之前檢查是否相同，不同的話就會 abort

- 有 canary 的話不能蓋到 return address、rbp

```
% ./test
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
*** stack smashing detected ***: <unknown>
terminated
zsh: abort (core dumped)  ./test
```



bof 應用

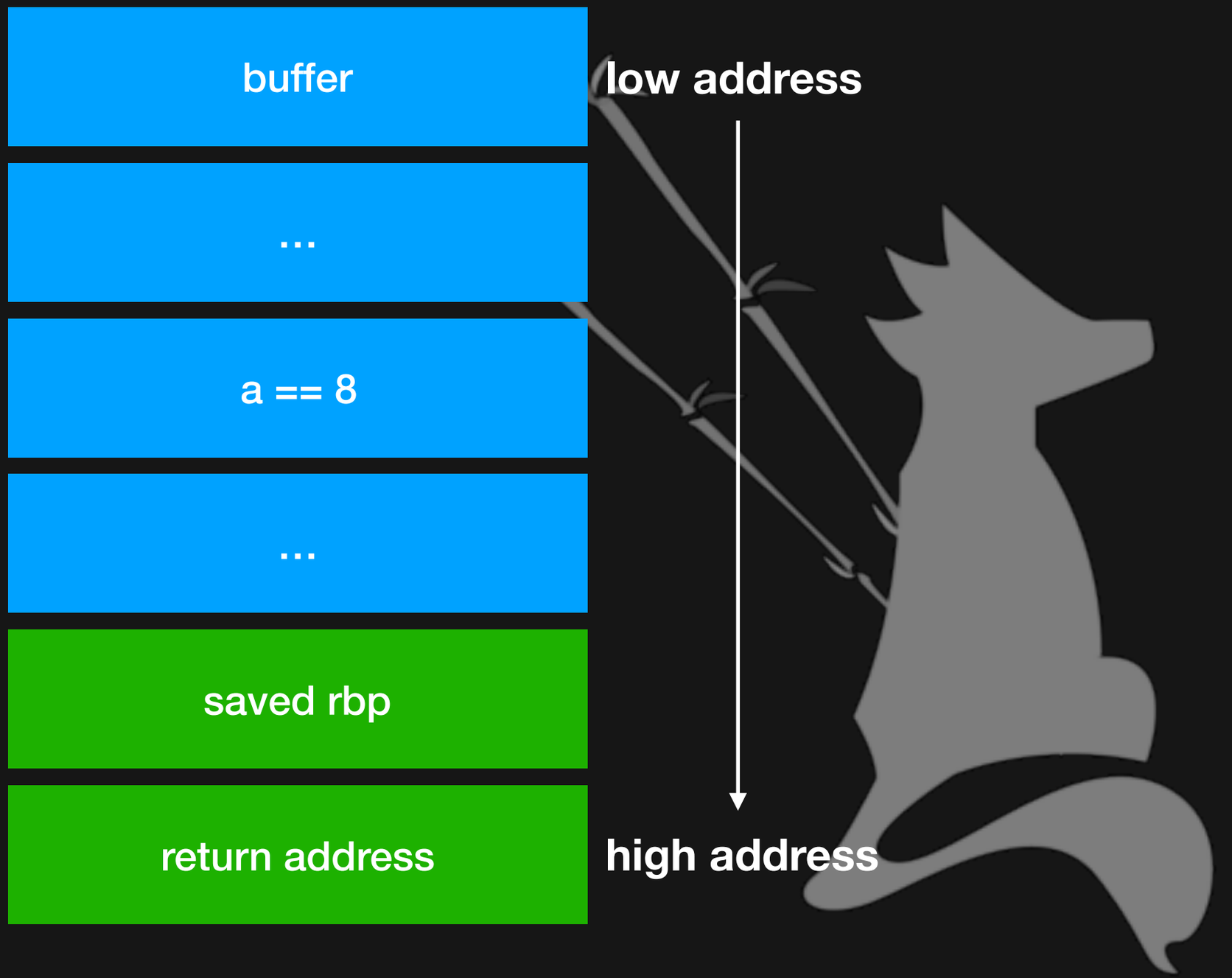
❖ 先看看 stack 上有什麼

- local variable
- saved rbp ———> stack migration
- return address ———> ret2 series

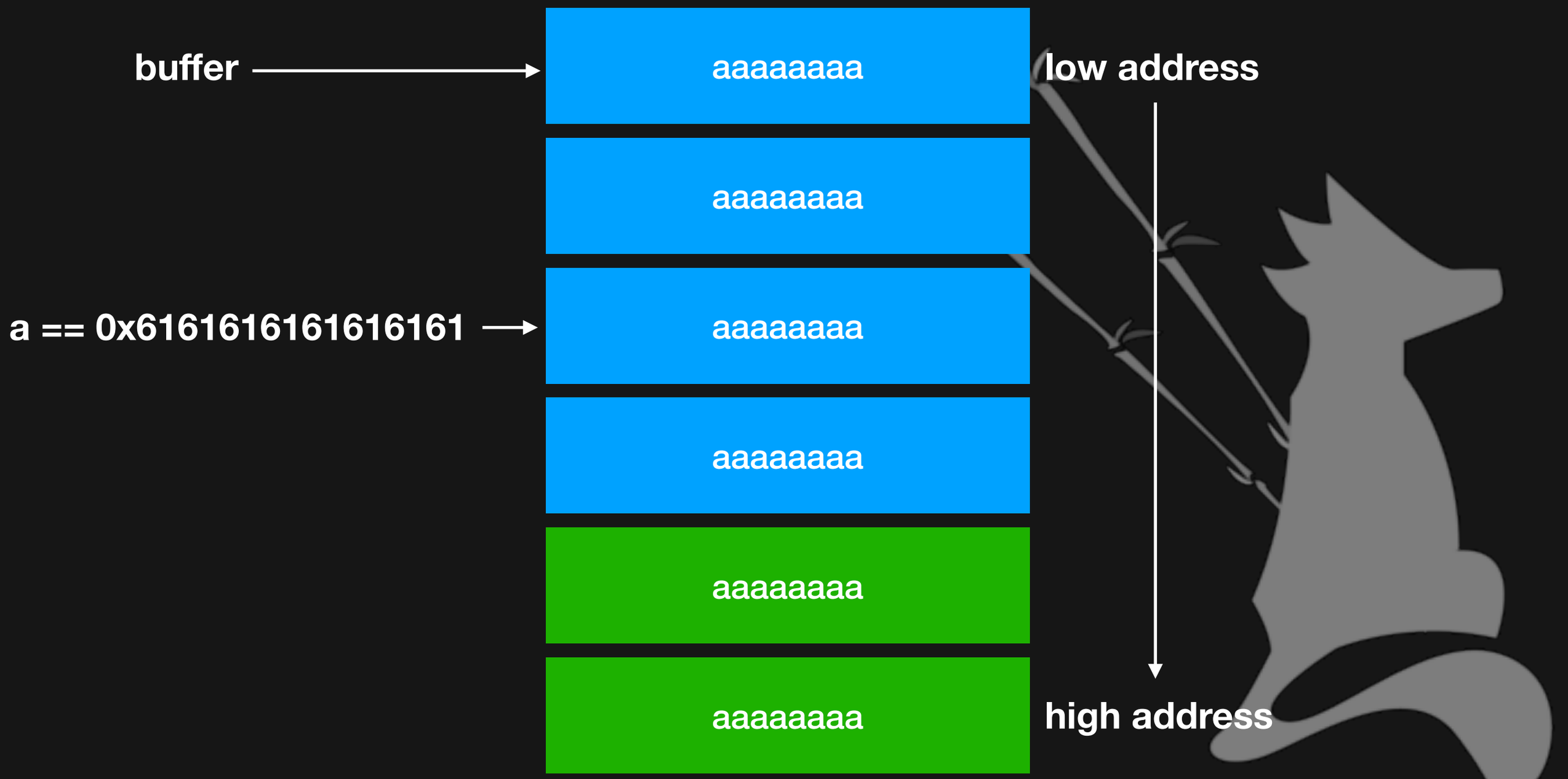
bof - local variable

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

int main(){
    int a = 8;
    char buffer[8];
    gets(buffer);

    if(a == 3){
        system(“/bin/sh”);
    }

    return 0;
}
```

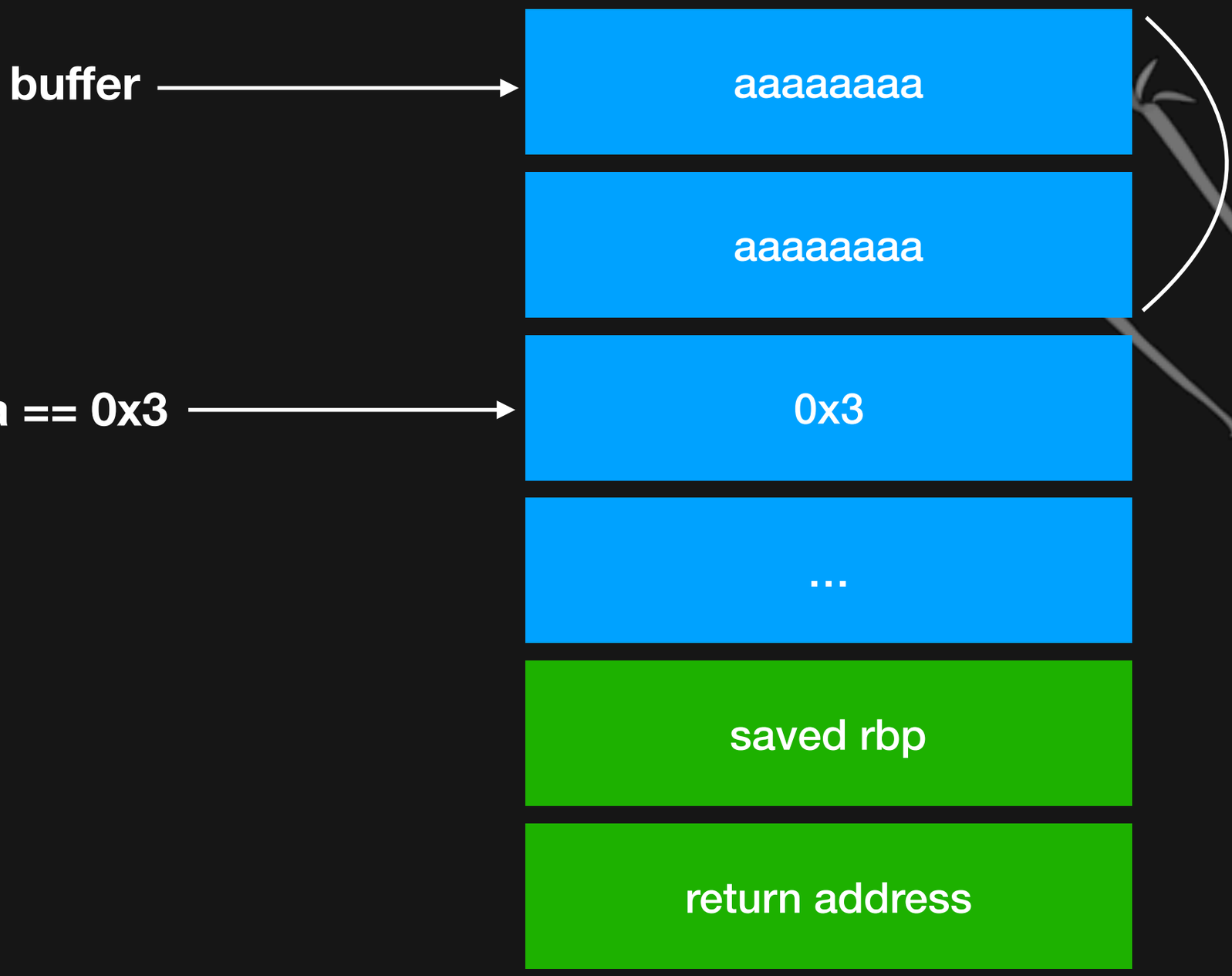
bof - local variable



bof - local variable



bof - local variable



如何算 offset ?

- 先隨意輸入來確定 buffer 位置
- 計算 buffer 位置和目標位置距離多遠

buffer

return address

```
0000| 0x7fffffffe790 --> 0x7fffffffe880 --> 0x1
0008| 0x7fffffffe798 ("AAAAAAAA")
0016| 0x7fffffffe7a0 --> 0x555555554600 (<register_tm_clones+16>:    sub    esi,edi)
0024| 0x7fffffffe7a8 --> 0x7ffff7a05b97 (<__libc_start_main+231>:    mov    edi,eax)
0032| 0x7fffffffe7b0 --> 0x1
```

```
offset = 0x7fffffffe7a8 - 0x7fffffffe798 = 0x10
```

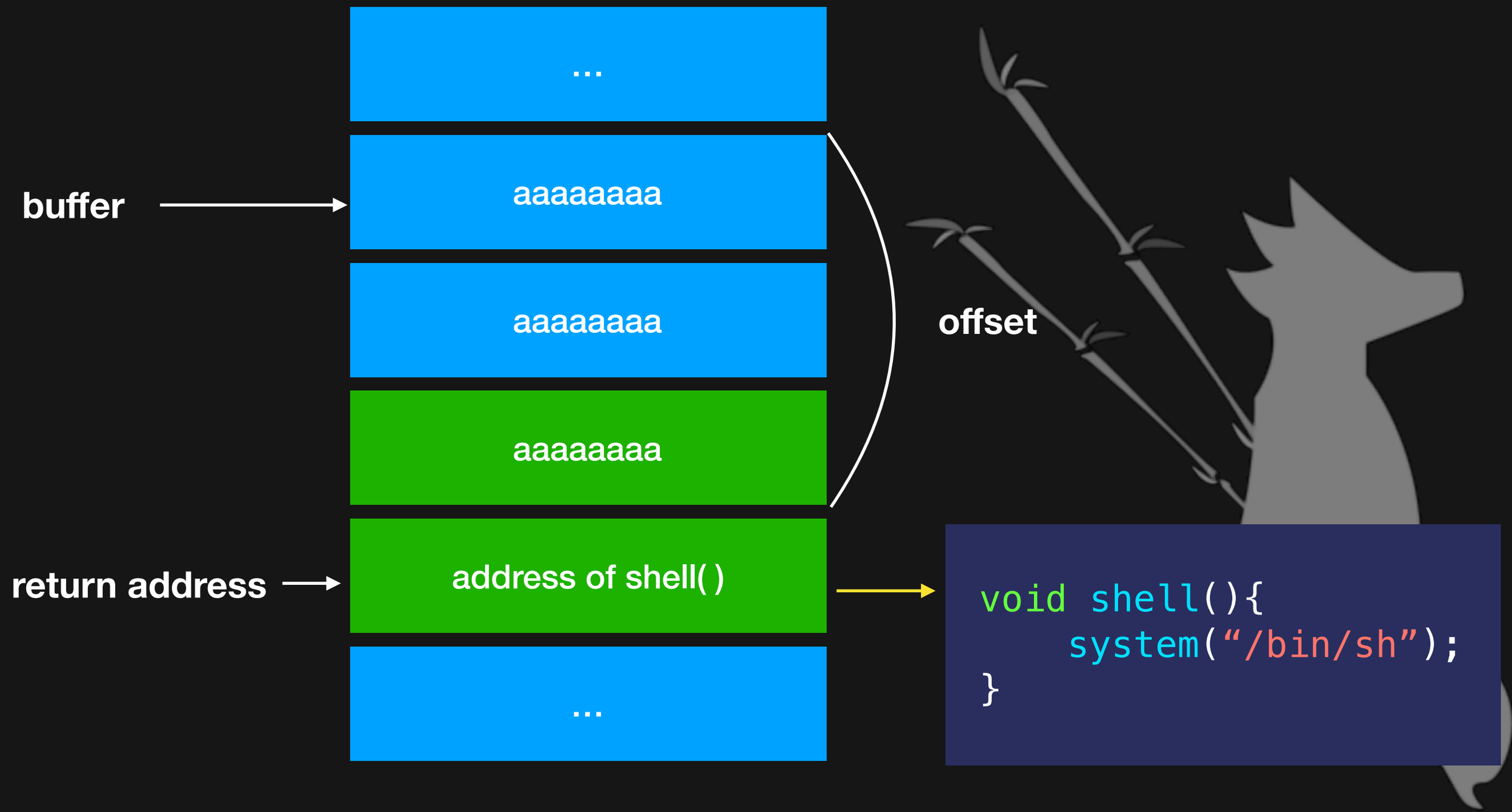
bof - ret2code

- 透過 Buffer Overflow 改變 return address
- 將 return address 改到 code 中任意處
- 須關閉 PIE

bof - ret2code

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

void shell(){
    system(“/bin/sh”);
}

int main(){
    char buffer[8];
    gets(buffer);
    return 0;
}
```

bof - ret2code



如何找到 address？

shell

```
% objdump -M intel -d test | less
```

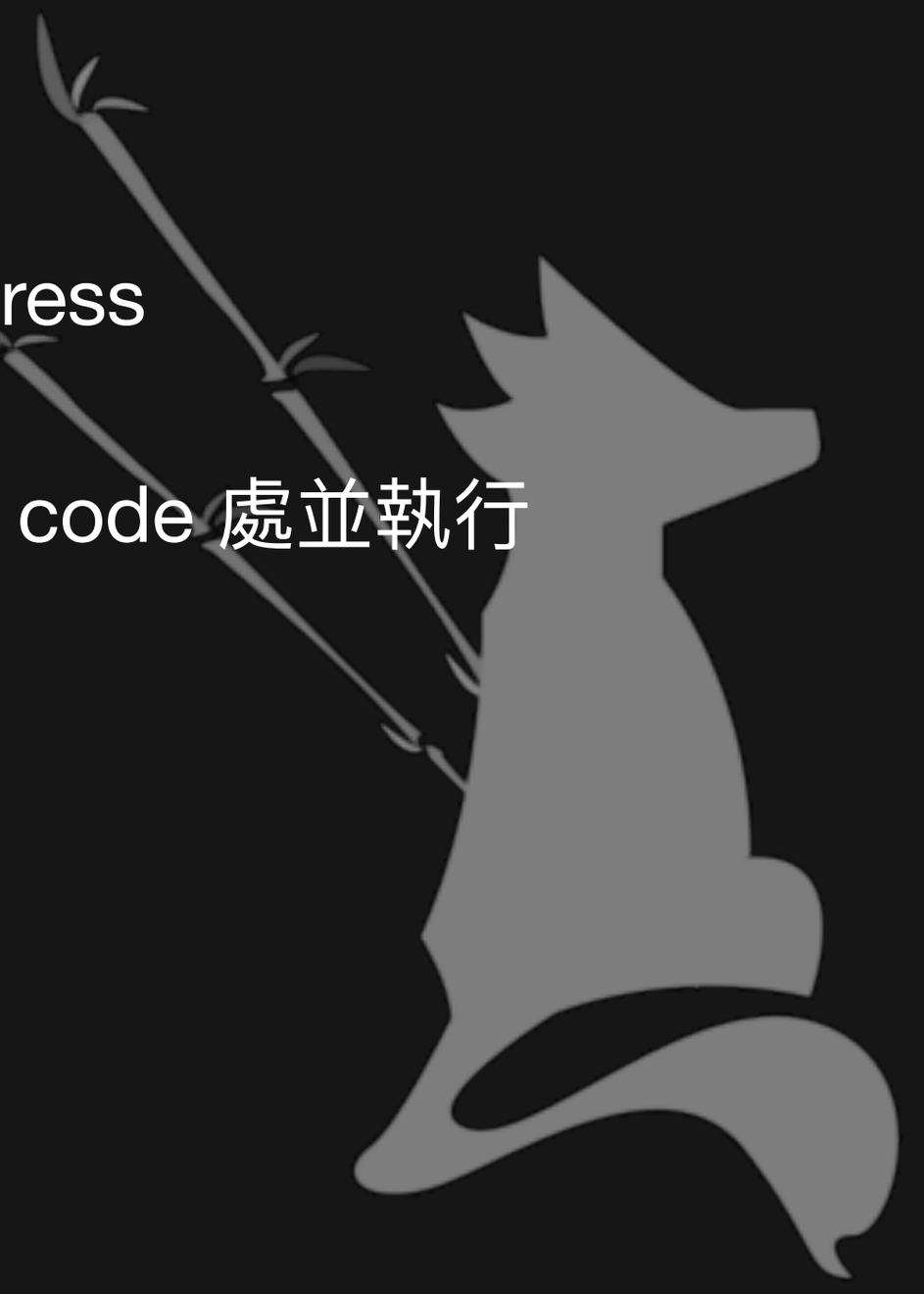
gdb

```
gdb-peda$ p shell
```

r2

```
[0x000005d0]> afl~shell
```

bof - ret2sc

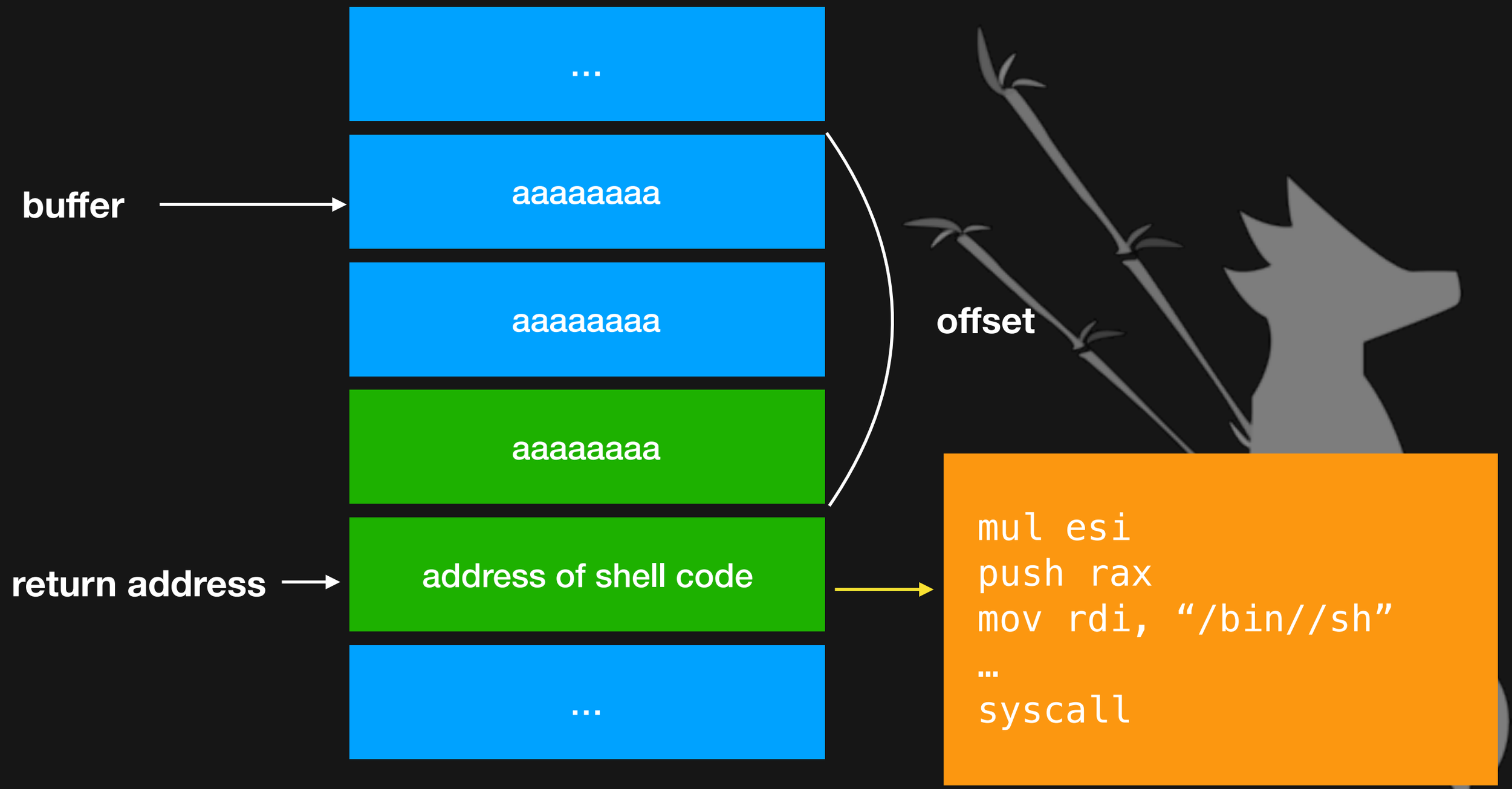


❖ 透過 Buffer Overflow 改變 return address

❖ 將 return address 改到自己寫的 shell code 處並執行

❖ 須關閉 NX

bof - ret2sc


...
aaaaaaaa
aaaaaaaa
aaaaaaaa
address of shell code
...
buffer
return address
offset
mul esi
push rax
mov rdi, “/bin//sh”
…
syscall

哪裡可以寫 shell code？

- 選擇含有 rwx 處
- 選擇中間部分，因為前後可能會有用到

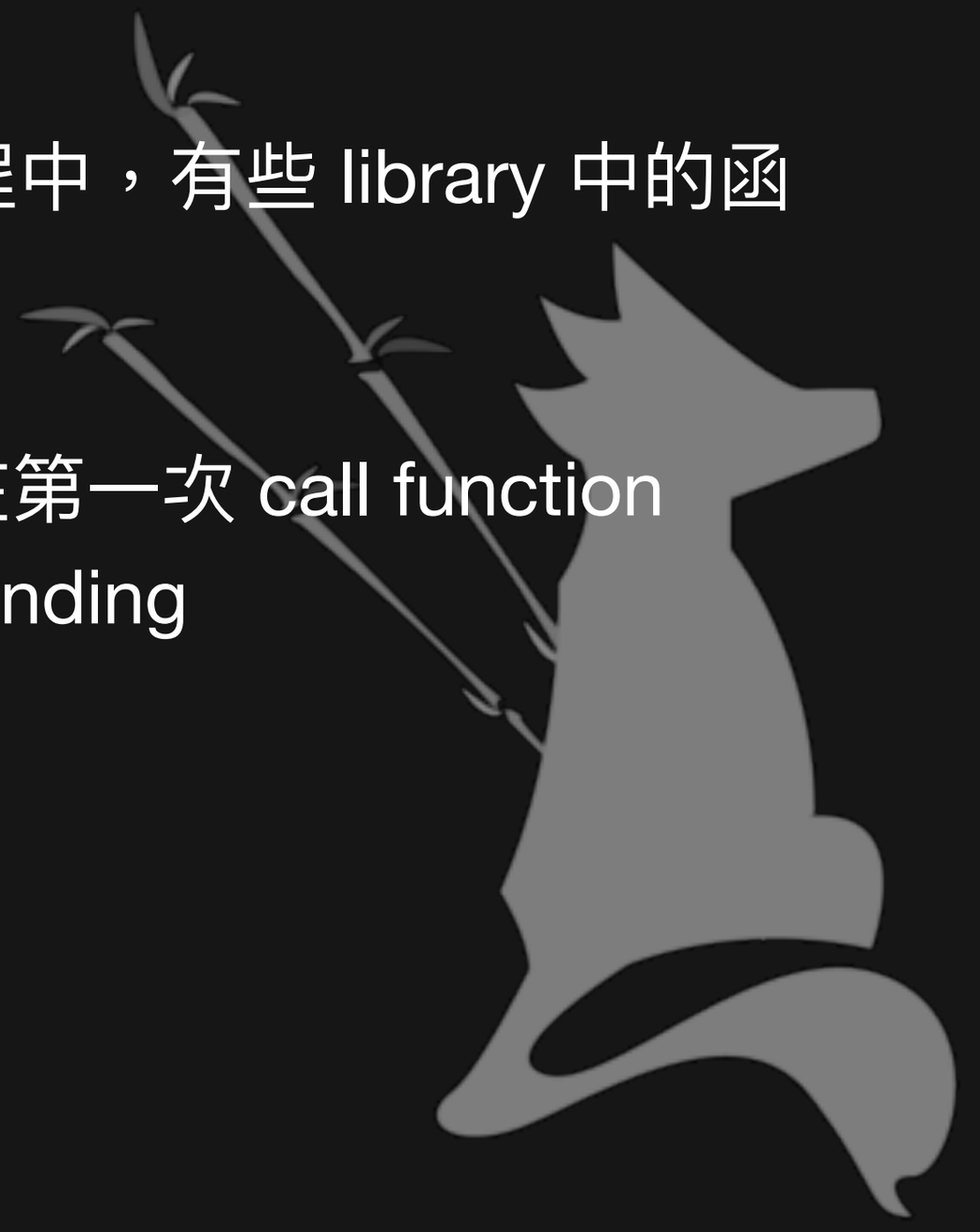
推薦

```
gdb-peda$ vmmap
Start              End                Perm      Name
0x00400000         0x00401000         r-xp      /home/frozenkp/csie/class/a.out
0x00600000         0x00601000         r-xp      /home/frozenkp/csie/class/a.out
0x00601000         0x00602000         rwxp      /home/frozenkp/csie/class/a.out
0x00007ffff79e4000 0x00007ffff7bcb000 r-xp      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7bcb000 0x00007ffff7dcb000 ---p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dcb000 0x00007ffff7dcf000 r-xp      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dcf000 0x00007ffff7dd1000 rwxp      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dd1000 0x00007ffff7dd5000 rwxp      mapped
0x00007ffff7dd5000 0x00007ffff7dfc000 r-xp      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7fe0000 0x00007ffff7fe2000 rwxp      mapped
0x00007ffff7ff7000 0x00007ffff7ffa000 r--p      [vvar]
0x00007ffff7ffa000 0x00007ffff7ffc000 r-xp      [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r-xp      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rwxp      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7ffe000 0x00007ffff7fff000 rwxp      mapped
0x00007ffffffde000 0x00007ffffffff000 rwxp      [stack]
0xffffffffff600000 0xffffffffff601000 r-xp      [vsyscall]
```

Libc

Lazy binding

- Dynamic linking 的程式在執行過程中，有些 library 中的函式可能到結束都不會執行到
- ELF 採取 Lazy binding 的機制，在第一次 call function 時，才會去尋找真正的位置進行 binding
- 詳細請參考 week2_concept



plt & got

- 因為 Lazy binding 的機制，當要用到 library 函式時，會 call 目標函式的 plt，接著才 call 目標函式的 got
- got 中存有目標函式在 library 中真正的位置

```
lea rax, [local_8h]
mov rdi, rax
; int puts(const char *s)
call sym.imp.puts;[gb]
```

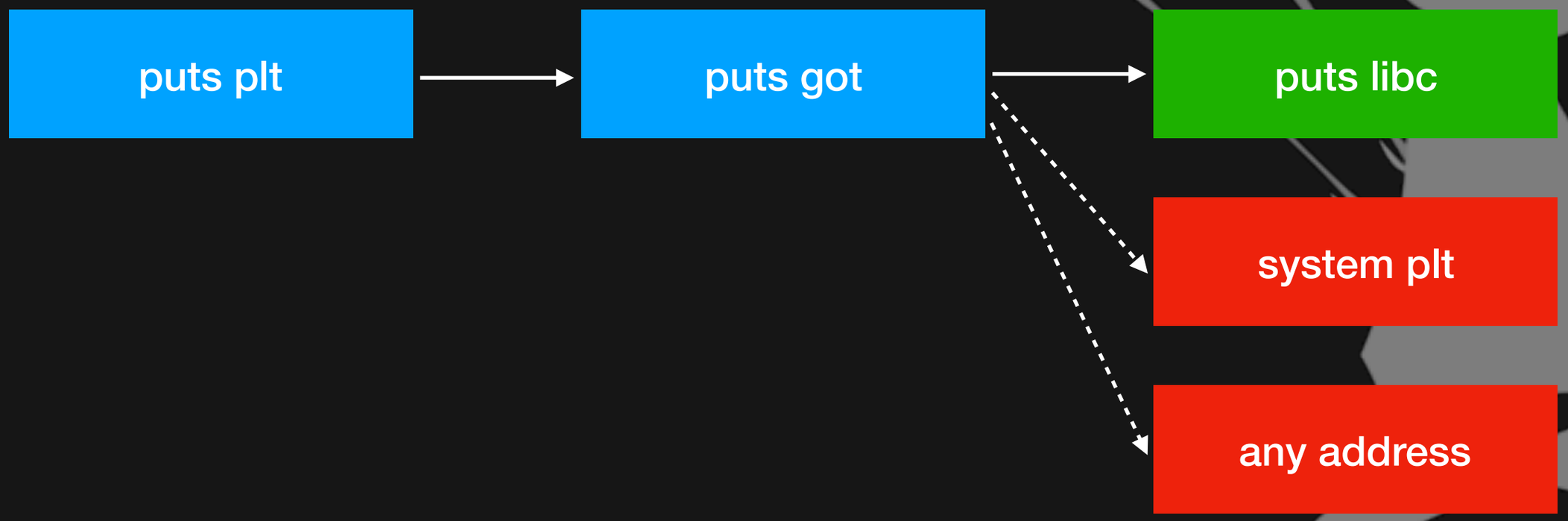
```
(fcn) sym.imp.puts 6
  sym.imp.puts (const char *s);
      ::    ; CALL XREF from sym.main (0x400557)
      ::    0x00400430      ff25e20b2000    jmp qword reloc.puts        ; [0x601018:8]=0x400436 ; "6\x04@"
      ::    0x00400436      6800000000      push 0
      `==<  0x0040043b      e9e0ffffff      jmp 0x400420
```

puts plt = 0x400430

puts got = 0x601018

GOT Hijacking

- 透過改寫 GOT 使得呼叫該函式時，跳到指定位置
- 不能是 Full RELRO



bof - ret2libc

- 通常 libc 中的函式並不會全部用到，但其中包含許多好用的函式
  - system
  - execve
- 因為 ASLR，libc 的位址會有 libc base，必須有 libc base 才可以使用 libc 函式

libc base 在哪裡？

❖ Libc base 只能在執行過程中 leak 出來

❖ 尋找執行中有用到 libc 的位址

- GOT 的內容
- stack 上的殘渣

libc base 在哪裡？

通常使用fmt string來leak stack

gets libc

__libc_start_main+231

```
gdb-peda$ x/gx 0x601020
0x601020:       0x00007ffff7a640b0
gdb-peda$ stack 10
0000| 0x7fffffffe740 --> 0x7fffffffe830 --> 0x1
0008| 0x7fffffffe748 --> 0x61 ('a')
0016| 0x7fffffffe750 --> 0x400570 (<__libc_csu_init>:   push   r15)
0024| 0x7fffffffe758 --> 0x7ffff7a05b97 (<__libc_start_main+231>:       mov    edi,eax)
0032| 0x7fffffffe760 --> 0x1
0040| 0x7fffffffe768 --> 0x7fffffffe838 --> 0x7fffffffeadf ("/home/frozenkp/csie/class/a.out")
0048| 0x7fffffffe770 --> 0x100008000
0056| 0x7fffffffe778 --> 0x400537 (<main>:      push   rbp)
0064| 0x7fffffffe780 --> 0x0
0072| 0x7fffffffe788 --> 0x808f706e6cc37e58
```

libc base 計算

https://youtu.be/oK3dKcyBSSA?t=4996

- 假設把 puts_got 印出來了
- puts_got_value = libc_base + puts_libc
  - libc_base = puts_got_value - puts_libc
  - system_got_value = libc_base + system_libc
- 如何取得 puts_libc ?

```
% readelf -a ./libc.so.6 | grep puts
```

one_gadget

- 在 libc 中可以一次 get shell 的位址
- 必須符合規定的限制，且擁有 libc base

```
[XD] % one_gadget /lib/x86_64-linux-gnu/libc.so.6
0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
constraints:
  rcx == NULL

0x4f322 execve("/bin/sh", rsp+0x40, environ)
constraints:
  [rsp+0x40] == NULL

0x10a38c        execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL
```

Return Oriented Programming

ROP

- Return Oriented Programming 簡稱 ROP
- 透過串接 Gadget 達成控制流程的目的

Gadget

❖ 結尾是 ret 的程式碼片段

```
pop rax
ret
```

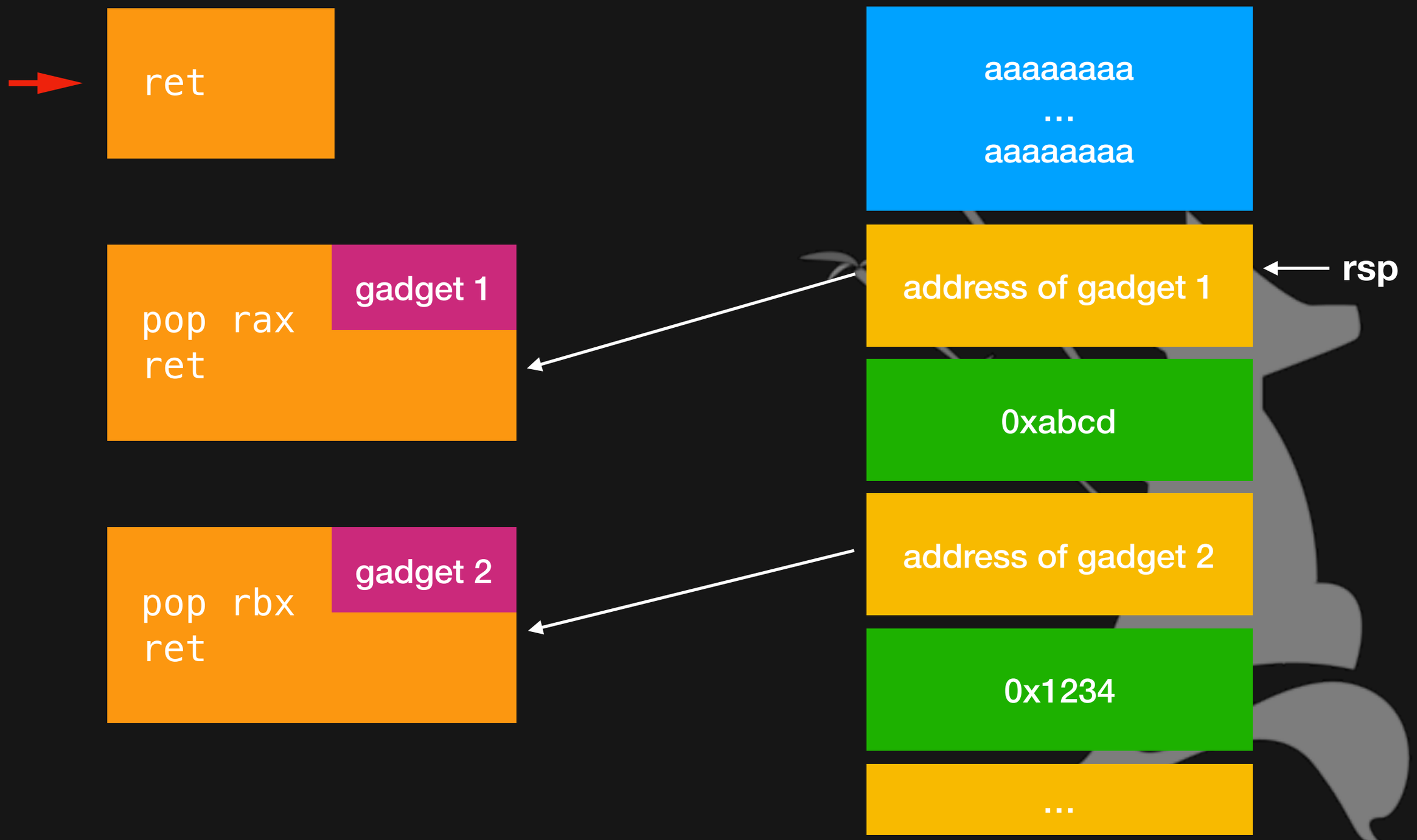
```
mov rax, rbx
ret
```

```
lea rax, [local_8h]
mov rdi, rax
mov eax, 0
call sym.imp.gets
ret
```

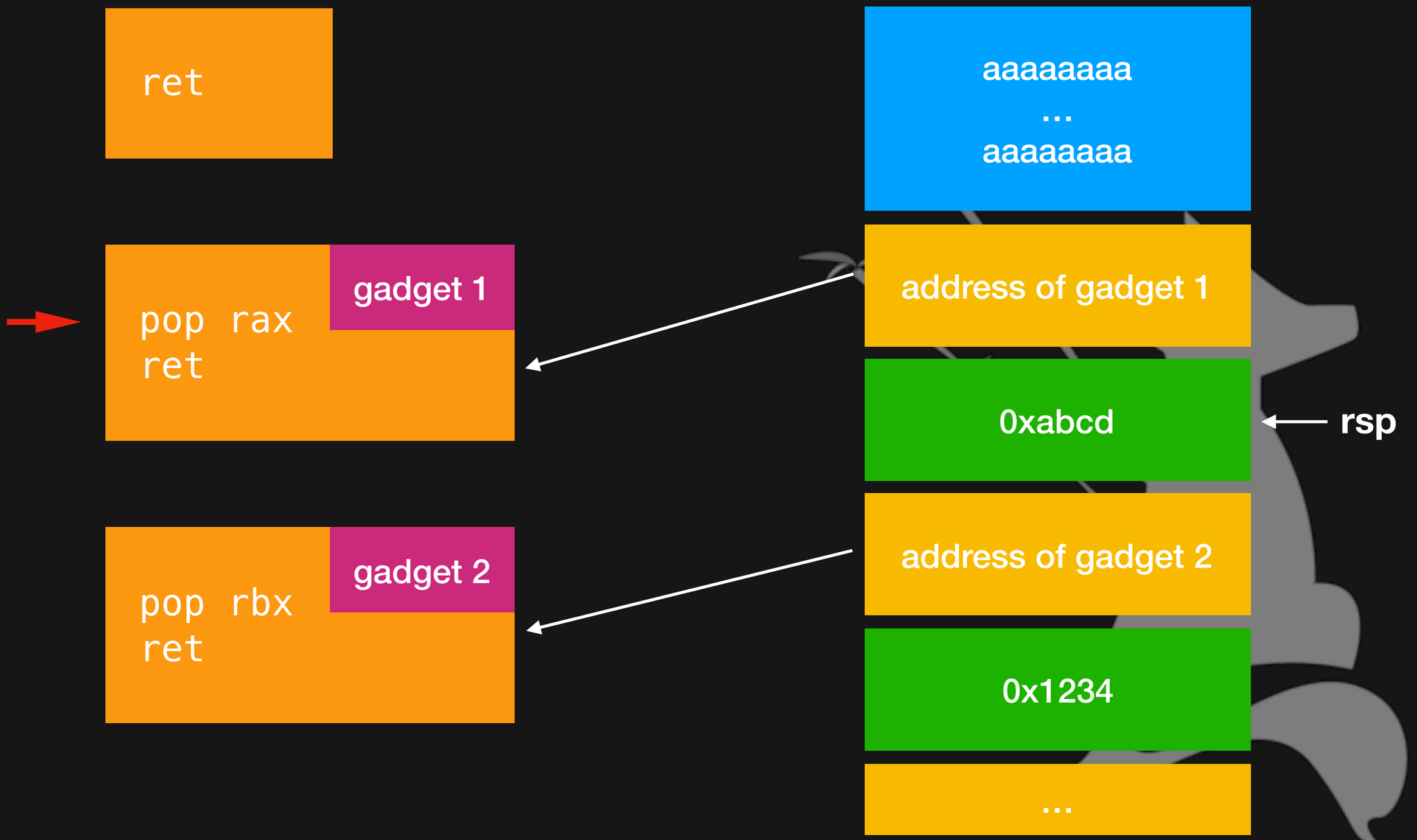
❖ 利用 ROPgadget 尋找適合的 gadget

```
% ROPgadget --binary ./test | grep ‘pop rax.*ret’
```

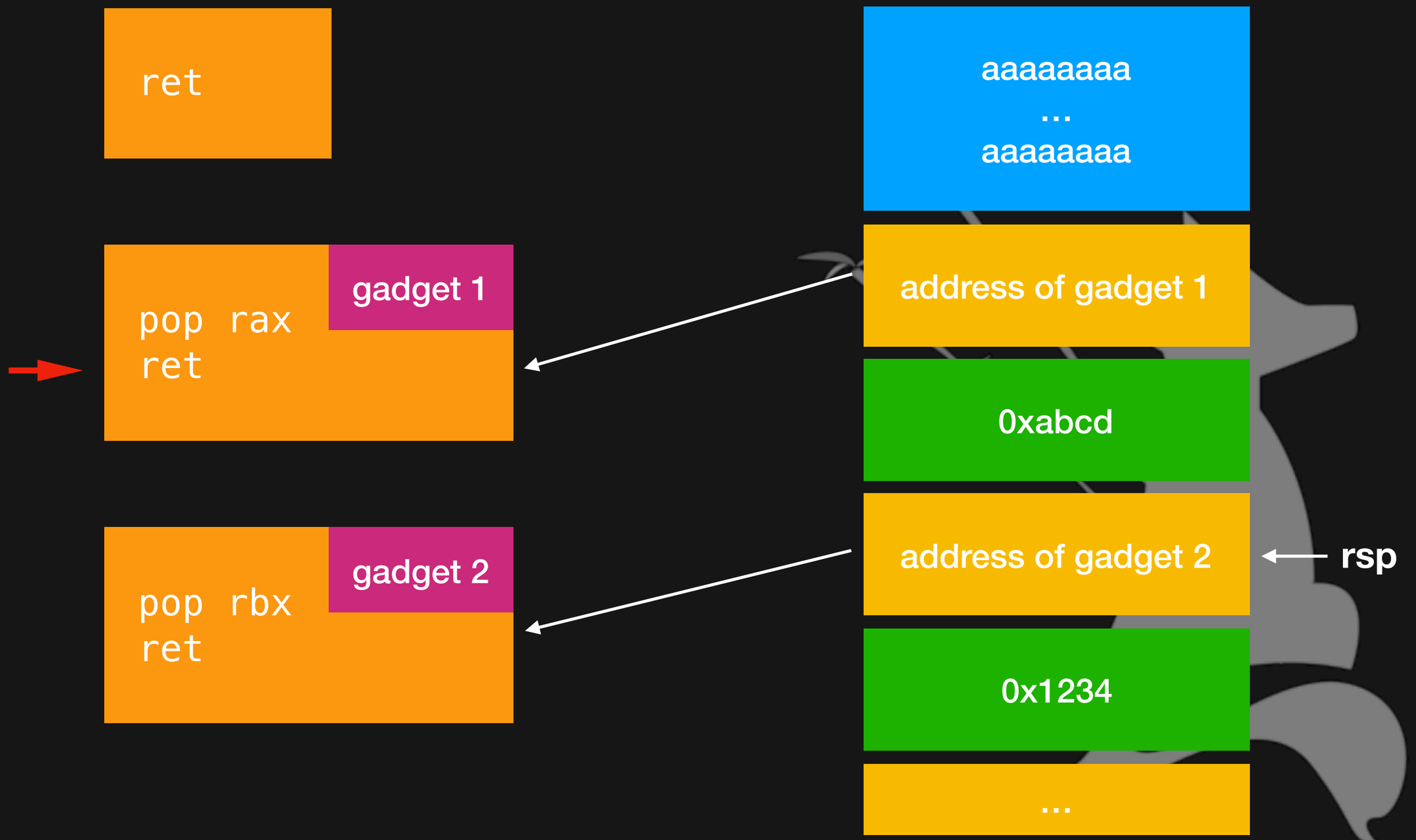
ROP chain


ret
aaaaaaaa
...
aaaaaaaa
address of gadget 1
rsp
pop rax
ret
gadget 1
0xabcd
address of gadget 2
pop rbx
ret
gadget 2
0x1234
...

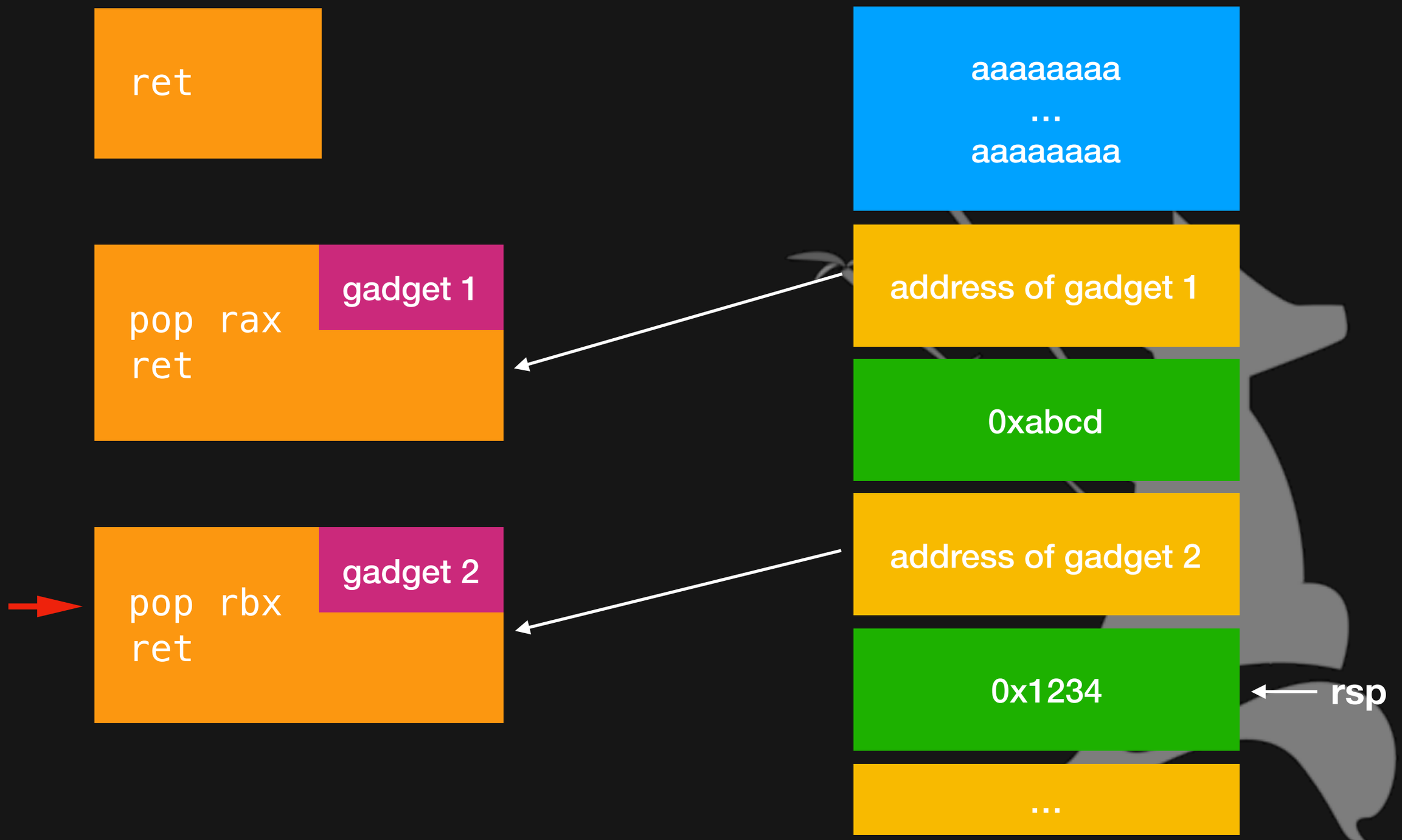
ROP chain


ret
aaaaaaaa
...
aaaaaaaa
pop rax
ret
gadget 1
address of gadget 1
0xabcd
rsp
pop rbx
ret
gadget 2
address of gadget 2
0x1234
...

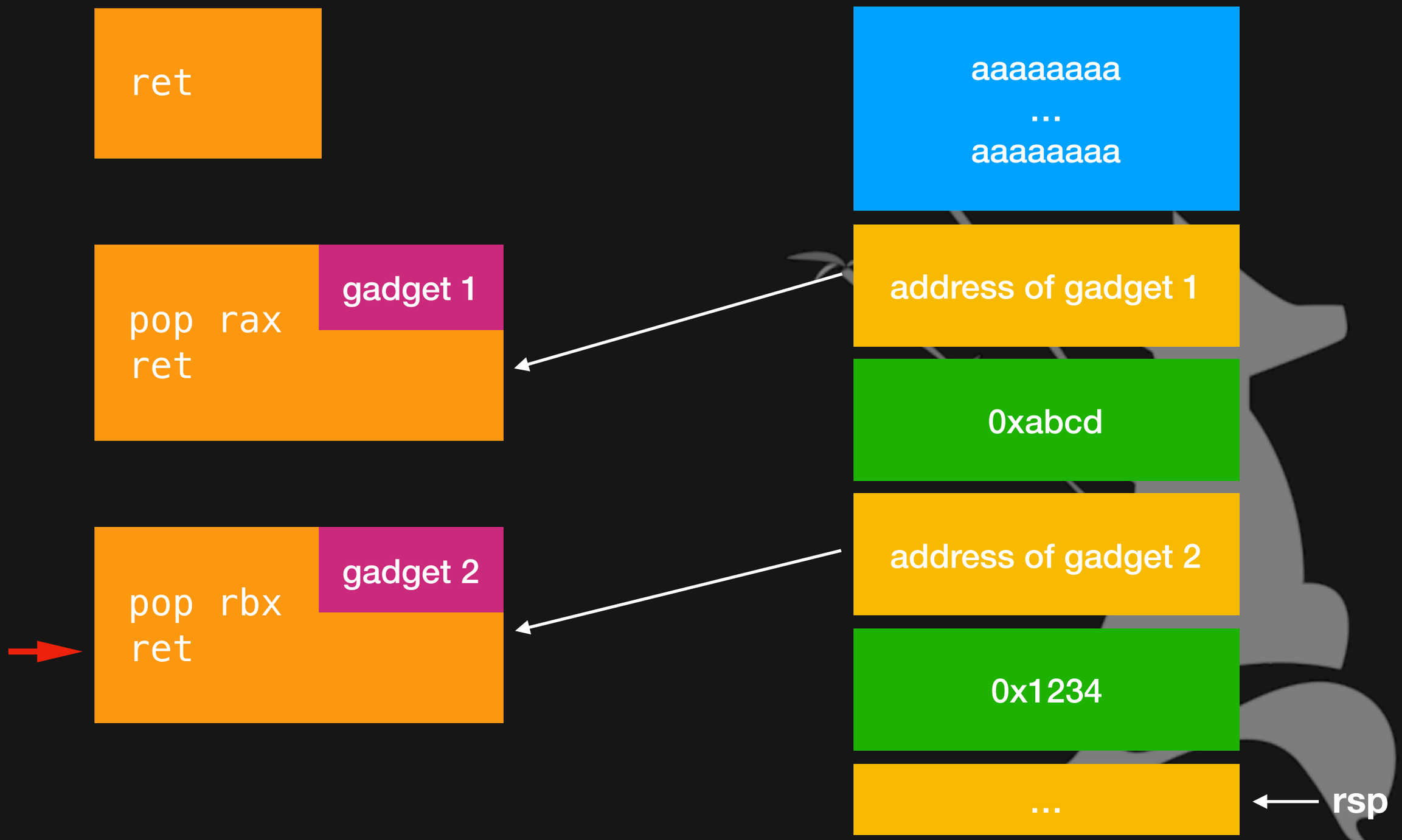
ROP chain


ret
aaaaaaaa
...
aaaaaaaa
gadget 1
pop rax
ret
address of gadget 1
0xabcd
gadget 2
pop rbx
ret
address of gadget 2
rsp
0x1234
...

ROP chain


ret
aaaaaaaa
...
aaaaaaaa
gadget 1
pop rax
ret
address of gadget 1
0xabcd
gadget 2
pop rbx
ret
address of gadget 2
0x1234
rsp
...

ROP chain


ret
aaaaaaaa
...
aaaaaaaa
gadget 1
pop rax
ret
address of gadget 1
0xabcd
gadget 2
pop rbx
ret
address of gadget 2
0x1234
...
rsp

execve(“/bin/sh”)

- x86_syscall
- rax = 0x3b
- rdi = address of ‘/bin/sh’
  - rdi -> buffer -> ‘/bin/sh’
- rsi = 0
- rdx = 0
- 設定好以後 call syscall

Reverse & Pwn on Go

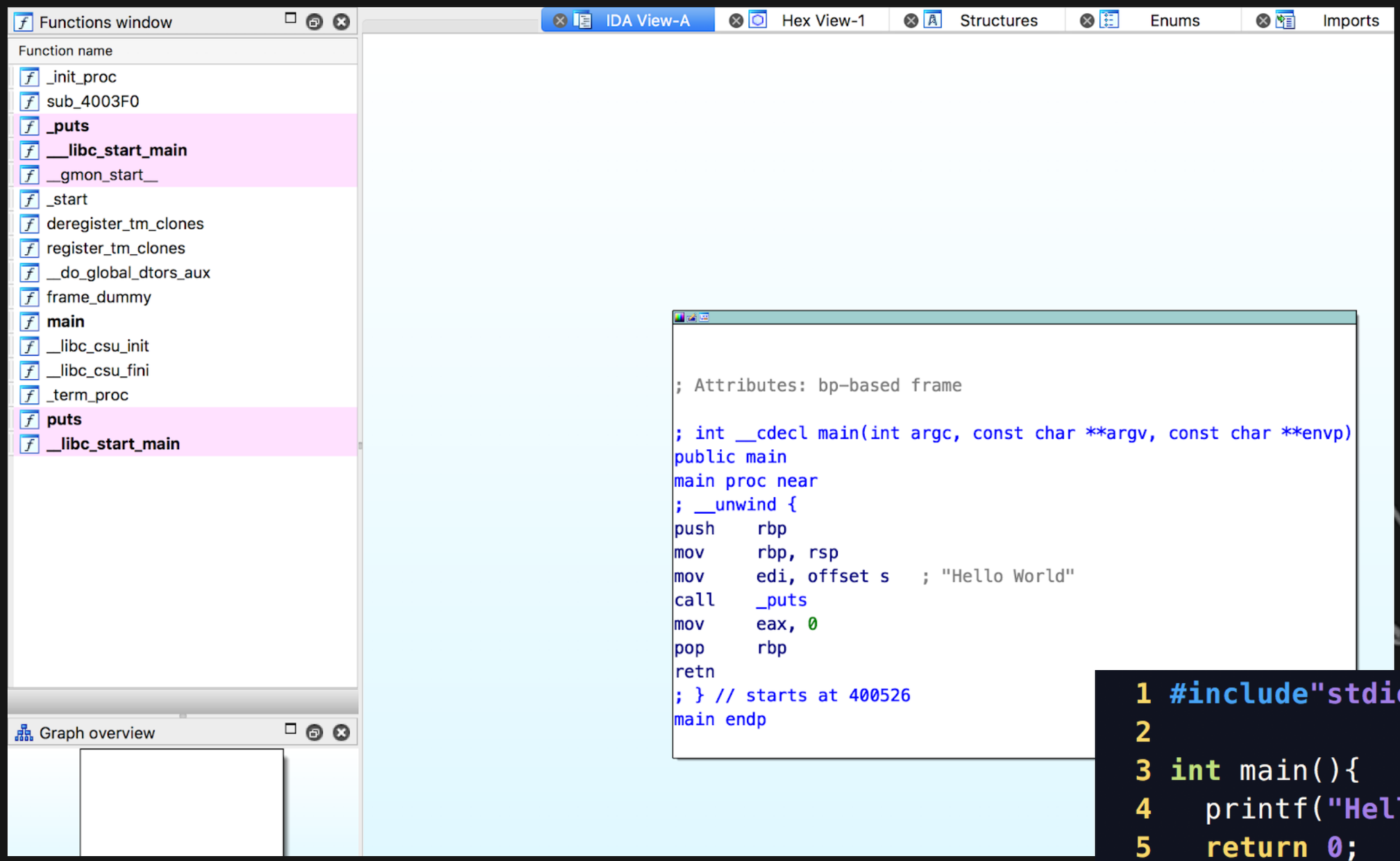
Why Go ?

- More big
- More complicated
- More malwares written in Go
  - GoBot
  - GoBot2
  - GoAT

Why Go ?

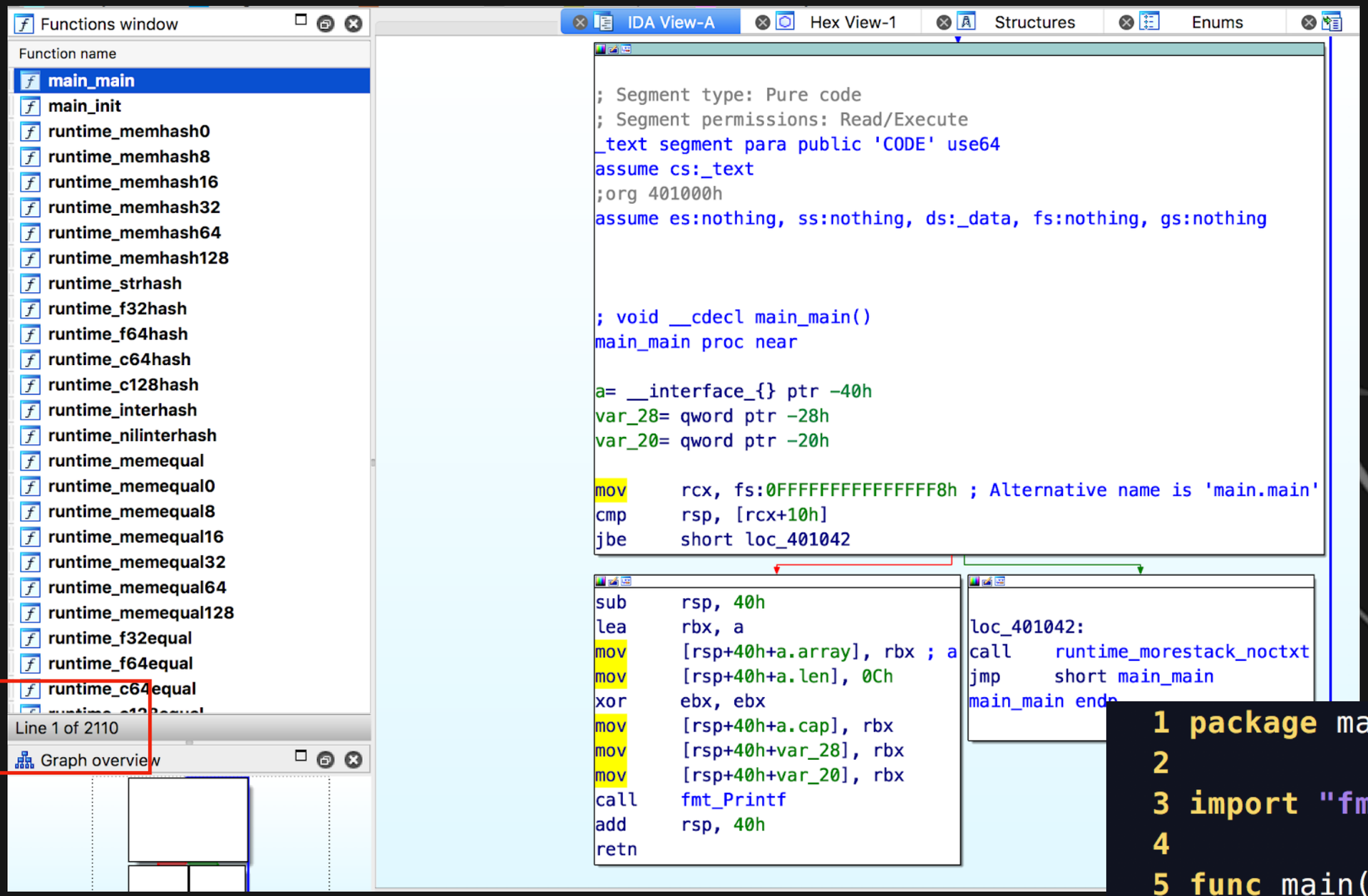
- More big
- More complicated
- More malwares written in Go
- Application in Go
  - Docker
  - Blockchain

Hello World in C




```c
#include"stdio.h"

int main(){
  printf("Hello World\n");
  return 0;
}

```

Hello World in Go

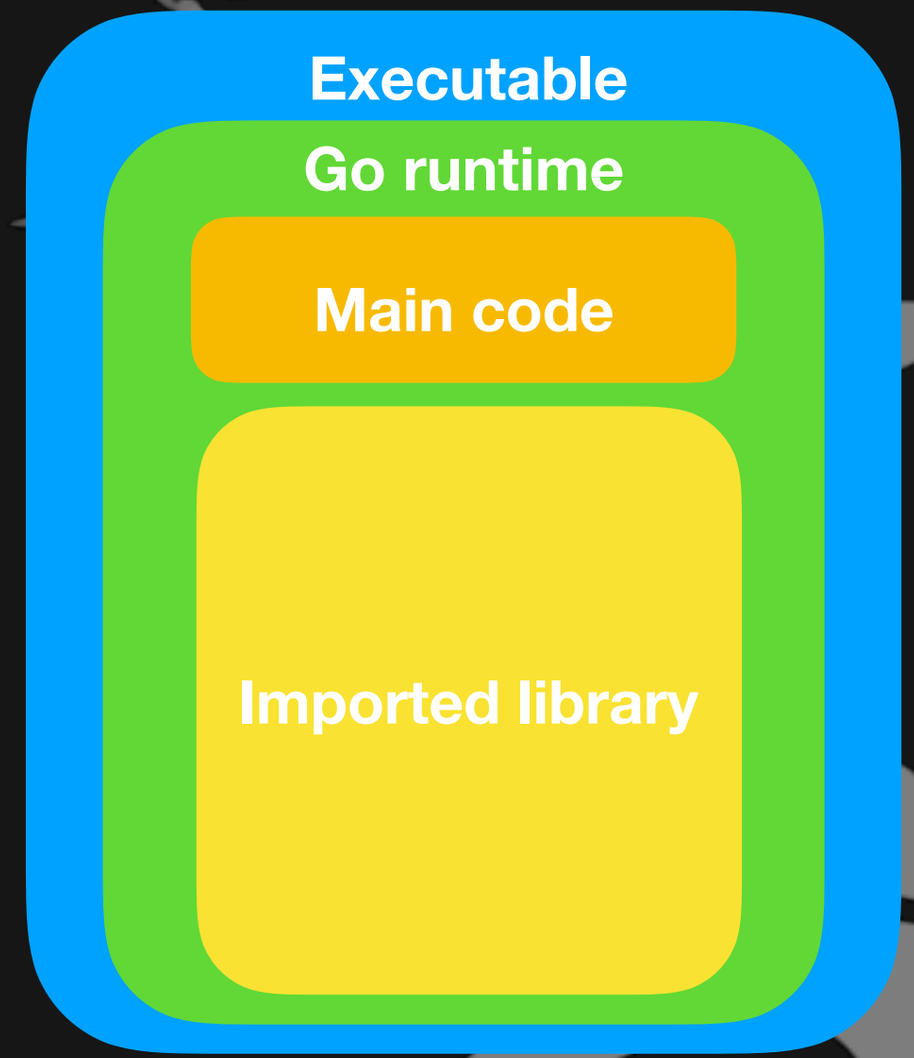



```go
package main

import "fmt"

func main(){
  fmt.Printf("Hello World\n")
}

```

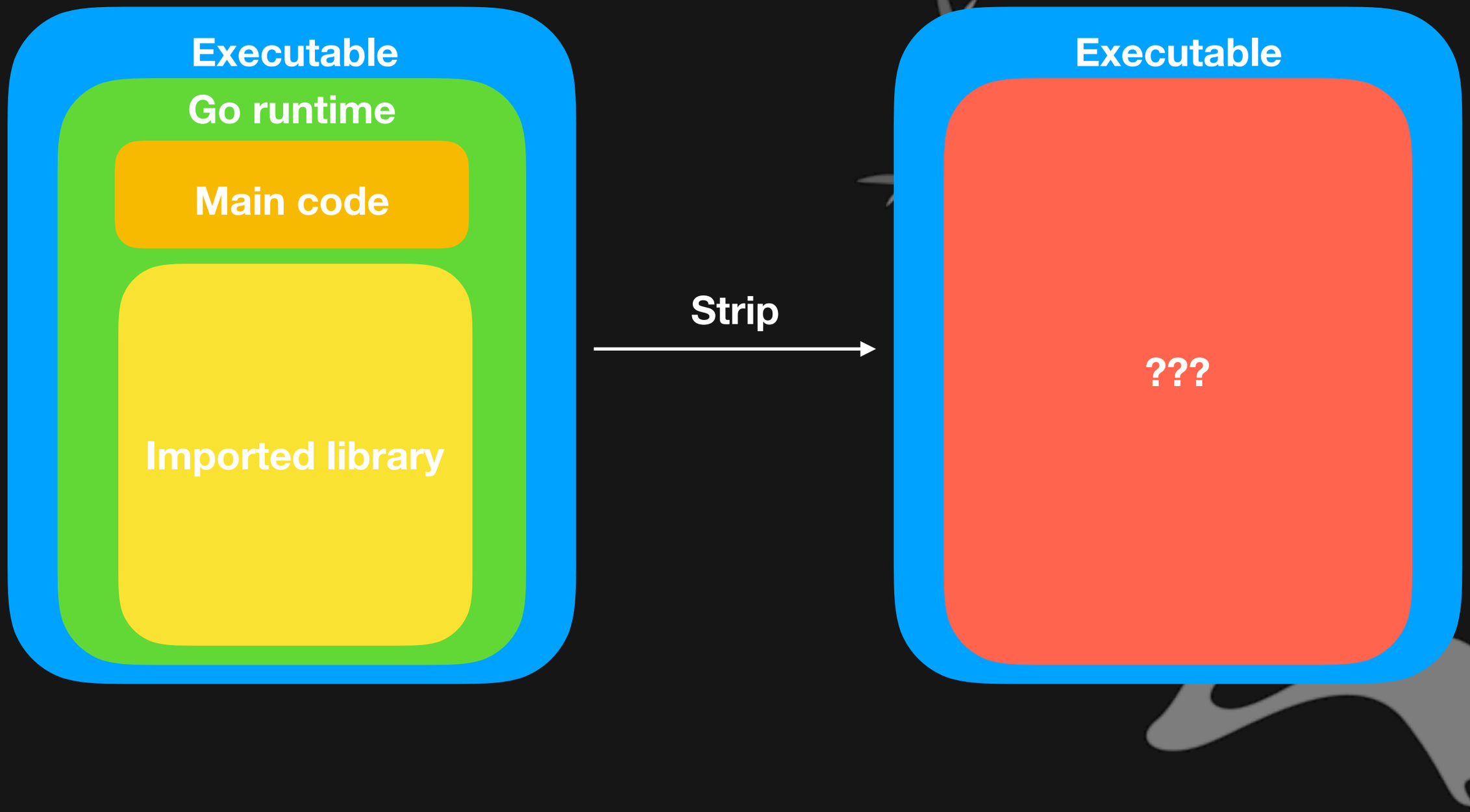
What’s in Go binary ?

❖ Take “Hello World” as example

- runtime: 911
- main: 2
- imported library: 1187



What’s in Go binary ?



Something interesting

```
[XD] % file hello
hello: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
21:52 frozenkp@Yi-Hsiende-MacBook-Pro(192.168.0.100)[~/Documents/pwn_docker/data/go/hello]
[XD] % strings hello | grep main
runtime.main not on m0
numerical argument out of domain
no goroutines (main called runtime.Goexit) - deadlock!
main.main
main.init
runtime.main
runtime.main.func1
runtime.main.func2
main
```

What’s in section ?

- .gosymtab (Null after go1.3)
- .gopclntab

```
04 0x0012f770      0 0x0052f770      0 -r-- .gosymtab
05 0x0012f780 328289 0x0052f780 328289 -r-- .gopclntab
```

.gopclntab

```
[0x0052f780 78% 2184 ./hello]> xc @ section..gopclntab
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0052f780  fbff ffff 0000 0108 3208 0000 0000 0000  ........2.......
0x0052f790  0010 4000 0000 0000 4083 0000 0000 0000  ..@.....@.......
0x0052f7a0  5010 4000 0000 0000 a883 0000 0000 0000  P.@.............
0x0052f7b0  a010 4000 0000 0000 0884 0000 0000 0000  ..@.............
0x0052f7c0  b010 4000 0000 0000 6084 0000 0000 0000  ..@.....`.......
0x0052f7d0  0011 4000 0000 0000 c884 0000 0000 0000  ..@.............
0x0052f7e0  5011 4000 0000 0000 3885 0000 0000 0000  P.@.....8.......
0x0052f7f0  a011 4000 0000 0000 a885 0000 0000 0000  ..@.............
0x0052f800  f011 4000 0000 0000 1886 0000 0000 0000  ..@.............
0x0052f810  4012 4000 0000 0000 8886 0000 0000 0000  @.@.............
0x0052f820  9012 4000 0000 0000 f886 0000 0000 0000  ..@.............
0x0052f830  6013 4000 0000 0000 7887 0000 0000 0000  `.@.....x.......
0x0052f840  3014 4000 0000 0000 f887 0000 0000 0000  0.@.............
0x0052f850  b014 4000 0000 0000 7088 0000 0000 0000  ..@.....p.......
0x0052f860  3015 4000 0000 0000 e888 0000 0000 0000  0.@.............
0x0052f870  c016 4000 0000 0000 8089 0000 0000 0000  ..@.............
```

section header

```
[0x0052f780 78% 2184 ./hello]> xc @ section..gopclntab
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0052f780  fbff ffff 0000 0108 3208 0000 0000 0000  ........2.......
0x0052f790  0010 4000 0000 0000 4083 0000 0000 0000  ..@.....@.......
0x0052f7a0  5010 4000 0000 0000 a883 0000 0000 0000  P.@.............
0x0052f7b0  a010 4000 0000 0000 0884 0000 0000 0000  ..@.............
0x0052f7c0  b010 4000 0000 0000 6084 0000 0000 0000  ..@.....`.......
0x0052f7d0  0011 4000 0000 0000 c884 0000 0000 0000  ..@.............
0x0052f7e0  5011 4000 0000 0000 3885 0000 0000 0000  P.@.....8.......
0x0052f7f0  a011 4000 0000 0000 a885 0000 0000 0000  ..@.............
0x0052f800  f011 4000 0000 0000 1886 0000 0000 0000  ..@.............
0x0052f810  4012 4000 0000 0000 8886 0000 0000 0000  @.@.............
0x0052f820  9012 4000 0000 0000 f886 0000 0000 0000  ..@.............
0x0052f830  6013 4000 0000 0000 7887 0000 0000 0000  `.@.....x.......
0x0052f840  3014 4000 0000 0000 f887 0000 0000 0000  0.@.............
0x0052f850  b014 4000 0000 0000 7088 0000 0000 0000  ..@.....p.......
0x0052f860  3015 4000 0000 0000 e888 0000 0000 0000  0.@.............
0x0052f870  c016 4000 0000 0000 8089 0000 0000 0000  ..@.............
```

size

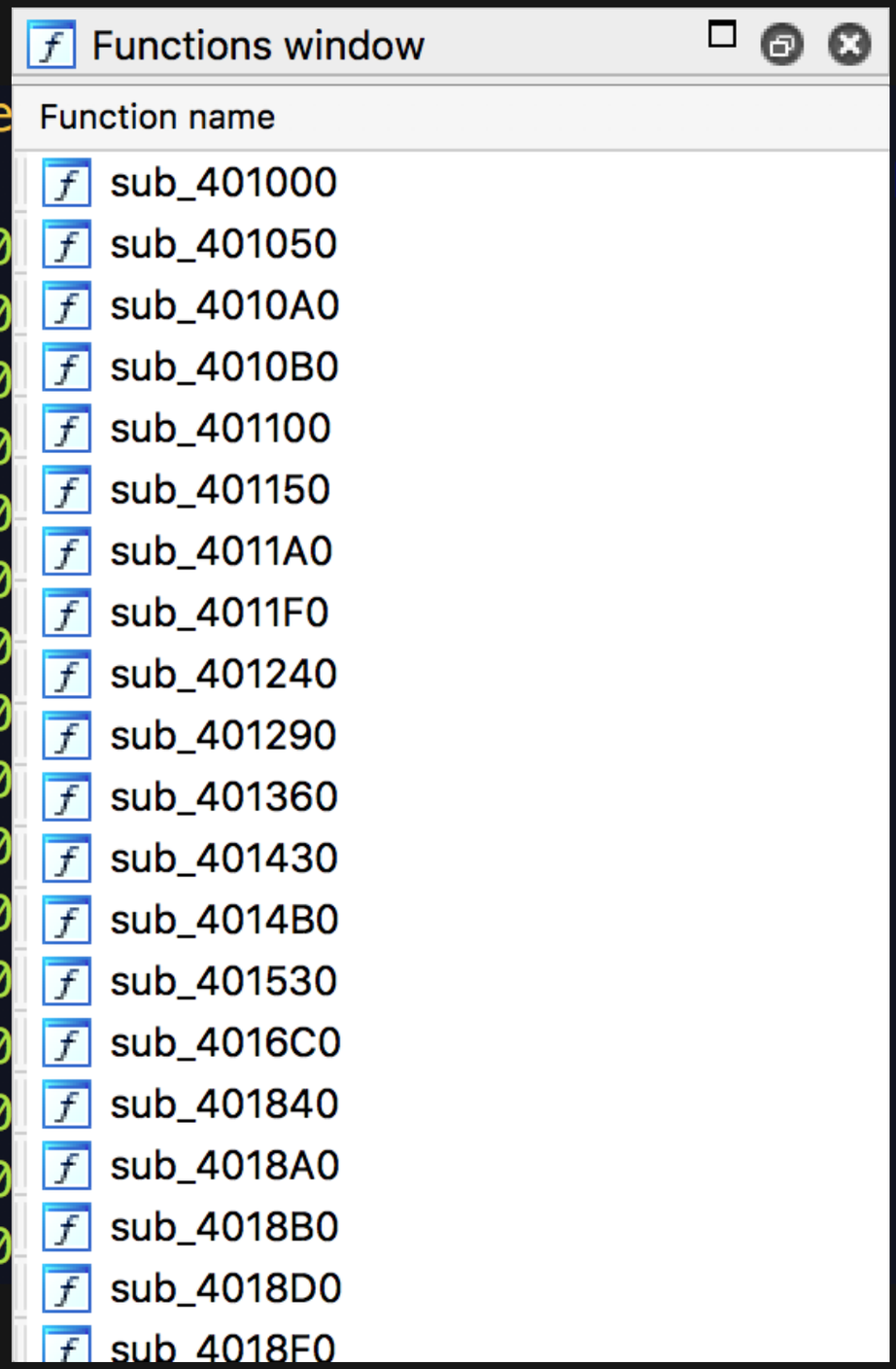
```
[0x0052f780 78% 2184 ./hello]> xc @ section..gopclntab
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0052f780  fbff ffff 0000 0108 3208 0000 0000 0000  ........2.......
0x0052f790  0010 4000 0000 0000 4083 0000 0000 0000  ..@.....@.......
0x0052f7a0  5010 4000 0000 0000 a883 0000 0000 0000  P.@.............
0x0052f7b0  a010 4000 0000 0000 0884 0000 0000 0000  ..@.............
0x0052f7c0  b010 4000 0000 0000 6084 0000 0000 0000  ..@.....`.......
0x0052f7d0  0011 4000 0000 0000 c884 0000 0000 0000  ..@.............
0x0052f7e0  5011 4000 0000 0000 3885 0000 0000 0000  P.@.....8.......
0x0052f7f0  a011 4000 0000 0000 a885 0000 0000 0000  ..@.............
0x0052f800  f011 4000 0000 0000 1886 0000 0000 0000  ..@.............
0x0052f810  4012 4000 0000 0000 8886 0000 0000 0000  @.@.............
0x0052f820  9012 4000 0000 0000 f886 0000 0000 0000  ..@.............
0x0052f830  6013 4000 0000 0000 7887 0000 0000 0000  `.@.....x.......
0x0052f840  3014 4000 0000 0000 f887 0000 0000 0000  0.@.............
0x0052f850  b014 4000 0000 0000 7088 0000 0000 0000  ..@.....p.......
0x0052f860  3015 4000 0000 0000 e888 0000 0000 0000  0.@.............
0x0052f870  c016 4000 0000 0000 8089 0000 0000 0000  ..@.............
```

function information

```
[0x0052f780 78% 2184 ./hello]> xc @ section..gopclntab
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0052f780  fbff ffff 0000 0108 3208 0000 0000 0000  ........2.......
0x0052f790  0010 4000 0000 0000 4083 0000 0000 0000  ..@.....@.......
0x0052f7a0  5010 4000 0000 0000 a883 0000 0000 0000  P.@.............
0x0052f7b0  a010 4000 0000 0000 0884 0000 0000 0000  ..@.............
0x0052f7c0  b010 4000 0000 0000 6084 0000 0000 0000  ..@.....`.......
0x0052f7d0  0011 4000 0000 0000 c884 0000 0000 0000  ..@.............
0x0052f7e0  5011 4000 0000 0000 3885 0000 0000 0000  P.@.....8.......
0x0052f7f0  a011 4000 0000 0000 a885 0000 0000 0000  ..@.............
0x0052f800  f011 4000 0000 0000 1886 0000 0000 0000  ..@.............
0x0052f810  4012 4000 0000 0000 8886 0000 0000 0000  @.@.............
0x0052f820  9012 4000 0000 0000 f886 0000 0000 0000  ..@.............
0x0052f830  6013 4000 0000 0000 7887 0000 0000 0000  `.@.....x.......
0x0052f840  3014 4000 0000 0000 f887 0000 0000 0000  0.@.............
0x0052f850  b014 4000 0000 0000 7088 0000 0000 0000  ..@.....p.......
0x0052f860  3015 4000 0000 0000 e888 0000 0000 0000  0.@.............
0x0052f870  c016 4000 0000 0000 8089 0000 0000 0000  ..@.............
```

function address

```
[0x0052f780 78% 2184 ./hello]> xc @ se
- offset -   0 1  2 3  4 5  6 7  8 9
0x0052f780  fbff ffff 0000 0108 3208 0
0x0052f790  0010 4000 0000 0000 4083 0
0x0052f7a0  5010 4000 0000 0000 a883 0
0x0052f7b0  a010 4000 0000 0000 0884 0
0x0052f7c0  b010 4000 0000 0000 6084 0
0x0052f7d0  0011 4000 0000 0000 c884 0
0x0052f7e0  5011 4000 0000 0000 3885 0
0x0052f7f0  a011 4000 0000 0000 a885 0
0x0052f800  f011 4000 0000 0000 1886 0
0x0052f810  4012 4000 0000 0000 8886 0
0x0052f820  9012 4000 0000 0000 f886 0
0x0052f830  6013 4000 0000 0000 7887 0
0x0052f840  3014 4000 0000 0000 f887 0
0x0052f850  b014 4000 0000 0000 7088 0
0x0052f860  3015 4000 0000 0000 e888 0
0x0052f870  c016 4000 0000 0000 8089 0
```



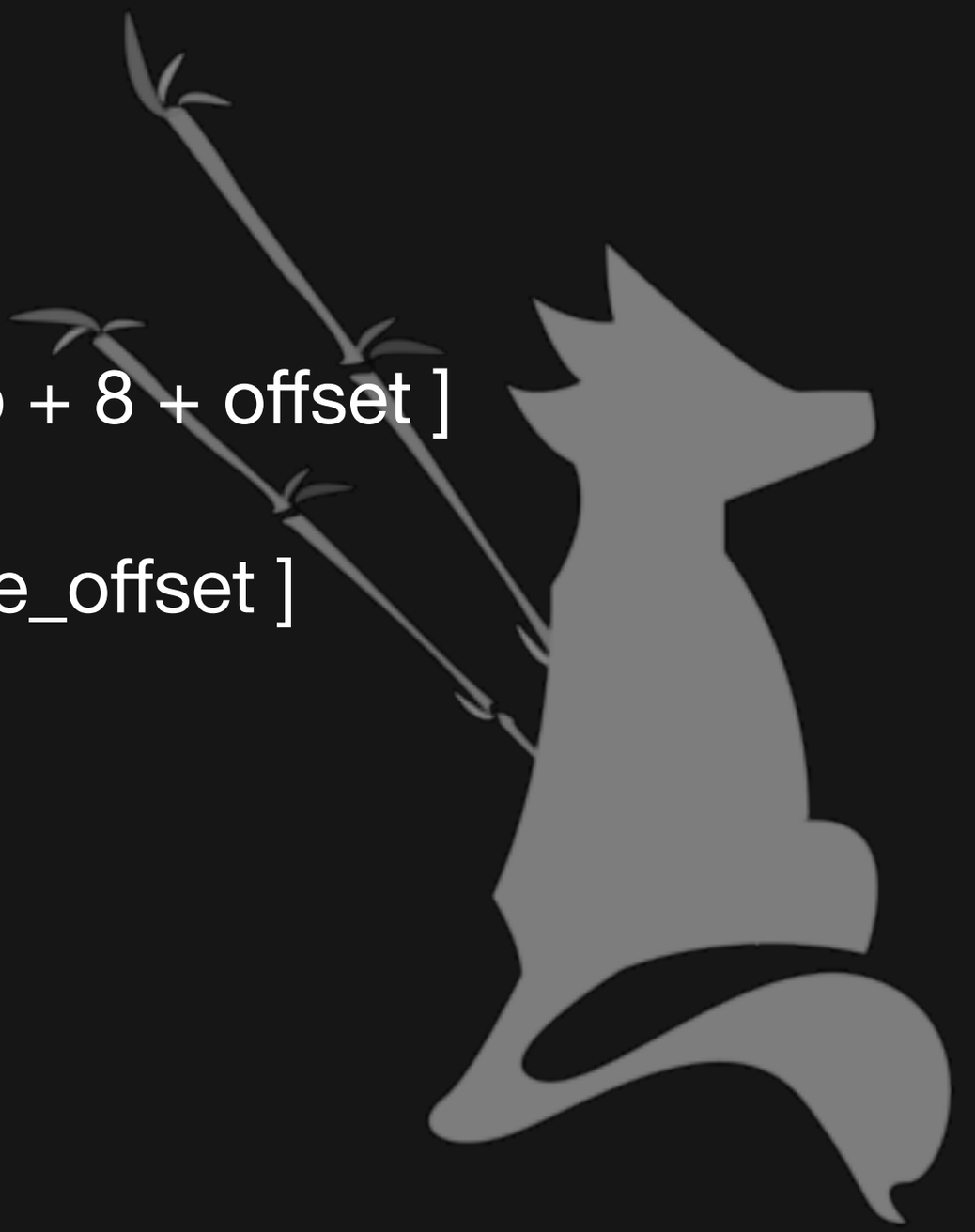
name offset

```
[0x0052f780 78% 2184 ./hello]> xc @ section..gopclntab
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0052f780  fbff ffff 0000 0108 3208 0000 0000 0000  ........2.......
0x0052f790  0010 4000 0000 0000 4083 0000 0000 0000  ..@.....@.......
0x0052f7a0  5010 4000 0000 0000 a883 0000 0000 0000  P.@.............
0x0052f7b0  a010 4000 0000 0000 0884 0000 0000 0000  ..@.............
0x0052f7c0  b010 4000 0000 0000 6084 0000 0000 0000  ..@.....`.......
```

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF  comment
0x00537b68  6d61 696e 2e69 6e69 7400 0002 5000 0450  main.init...P..P
0x00537b78  0010 5000 0028 0228 0000 0000 0000 0000  ..P..(.(........
0x00537b88  a010 4000 0000 0000 4084 0000 1800 0000  ..@.....@.......
0x00537b98  6745 2301 5284 0000 5584 0000 5884 0000  gE#.R...U...X...
0x00537ba8  0000 0000 0200 0000 8857 5200 0000 0000  .........WR.....
0x00537bb8  3057 5200 0000 0000 7275 6e74 696d 652e  0WR.....runtime.
0x00537bc8  6d65 6d68 6173 6830 0000 0210 0006 1000  memhash0........
0x00537bd8  7210 0000 0000 0000 b010 4000 0000 0000  r.........@.....
0x00537be8  a084 0000 1800 0000 6745 2301 b284 0000  ........gE#.....
0x00537bf8  b984 0000 bc84 0000 0100 0000 0200 0000  ................
0x00537c08  c384 0000 0000 0000 8857 5200 0000 0000  .........WR.....
0x00537c18  3057 5200 0000 0000 7275 6e74 696d 652e  0WR.....runtime.
0x00537c28  6d65 6d68 6173 6838 0000 0213 402f 3f0e  memhash8....@/?.
```

offset ?

- name_offset = (dword)[.gopclntab + 8 + offset]
- name = (string)[.gopclntab + name_offset]



舉個🌰

- .gopclntab = 0x0052f780
- func_addr = 0x4010b0
- offset = 0x8460

```
[0x0052f780 78% 2240 hello]> xc @ section..gopclntab
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0052f780  fbff ffff 0000 0108 3208 0000 0000 0000  ........2.......
0x0052f790  0010 4000 0000 0000 4083 0000 0000 0000  ..@.....@.......
0x0052f7a0  5010 4000 0000 0000 a883 0000 0000 0000  P.@.............
0x0052f7b0  a010 4000 0000 0000 0884 0000 0000 0000  ..@.............
0x0052f7c0  b010 4000 0000 0000 6084 0000 0000 0000  ..@.....`.......
0x0052f7d0  0011 4000 0000 0000 c884 0000 0000 0000  ..@.............
0x0052f7e0  5011 4000 0000 0000 3885 0000 0000 0000  P.@.....8.......
0x0052f7f0  a011 4000 0000 0000 a885 0000 0000 0000  ..@.............
0x0052f800  f011 4000 0000 0000 1886 0000 0000 0000  ..@.............
0x0052f810  4012 4000 0000 0000 8886 0000 0000 0000  @.@.............
```

舉個🌰

- offset + 0x8 + .gopclntab = 0x537be8
- name_offset = 0x84a0

```
[0x00537be8 80% 2184 hello]> xc @ entry0+924472 # 0x537be8
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00537be8  a084 0000 1800 0000 6745 2301 b284 0000  ........gE#.....
0x00537bf8  b984 0000 bc84 0000 0100 0000 0200 0000  ................
0x00537c08  c384 0000 0000 0000 8857 5200 0000 0000  .........WR.....
0x00537c18  3057 5200 0000 0000 7275 6e74 696d 652e  0WR.....runtime.
0x00537c28  6d65 6d68 6173 6838 0000 0213 402f 3f0e  memhash8....@/?.
0x00537c38  0006 5000 7613 0230 010d 0000 2f02 2100  ..P.v..0..../.!.
0x00537c48  0011 4000 0000 0000 0885 0000 1800 0000  ..@.............
0x00537c58  6745 2301 1b85 0000 2285 0000 2585 0000  gE#....."...%...
```

舉個🌰

- .gopclntab + name_offset = 0x537c20
- name = “runtime.memhash8”

- 0x4010b0 -> “runtime.memhash8”

```
[0x00537c20 80% 2184 hello]> xc @ entry0+924528 # 0x537c20
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00537c20  7275 6e74 696d 652e 6d65 6d68 6173 6838  runtime.memhash8
0x00537c30  0000 0213 402f 3f0e 0006 5000 7613 0230  ....@/?...P.v..0
0x00537c40  010d 0000 2f02 2100 0011 4000 0000 0000  ..../.!...@.....
0x00537c50  0885 0000 1800 0000 6745 2301 1b85 0000  ........gE#.....
0x00537c60  2285 0000 2585 0000 0100 0000 0200 0000  "...%...........
0x00537c70  2c85 0000 0000 0000 8857 5200 0000 0000  ,........WR.....
```

Pwnable ?

- bufio.Scanner / fmt.Scanf
- 在設計上不使用宣告好的 [ ]byte
- string 每次修改後皆換位置

Buffer Overflow

❖ 刻意製造的 Buffer Overflow

❖ unsafe pointer 相當於是 void pointer

❖ 當利用 unsafe pointer 寫入時，超出陣列範圍不會出錯

ROP gadget

- 為了可攜帶性，執行檔是 static link
- 包含極大量 gadget

Thanks for listening.