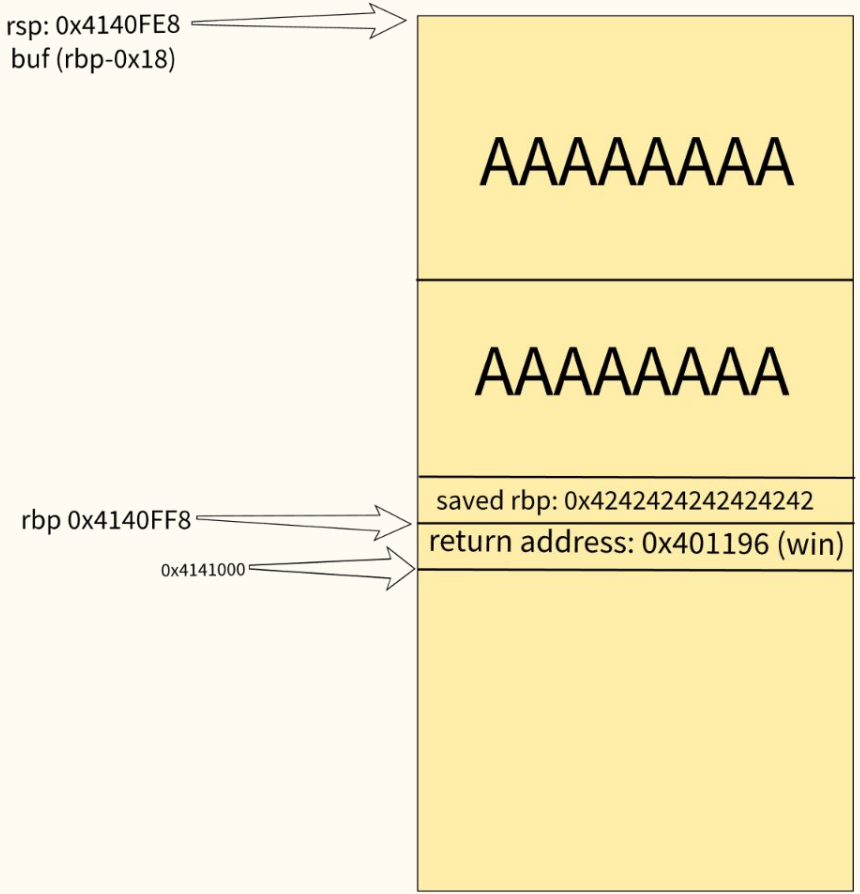
Return Oriented Programming

Return-Oriented Programming: because who needs function calls when you can just **Frankenstein** your way to shell access using the scraps of someone else's code?


You have already done single gadget rop chains.


rsp: 0x4140FE8
buf (rbp-0x18)
AAAAAAAA
AAAAAAAA
saved rbp: 0x4242424242424242
rbp 0x4140FF8
return address: 0x401196 (win)
0x4141000

What is a gadget?

- A single step in the chain.
- Typically only single exit.
- ret; jmp; call

For example:

```
0x41a83d
   xor r15, r15
   ret
```

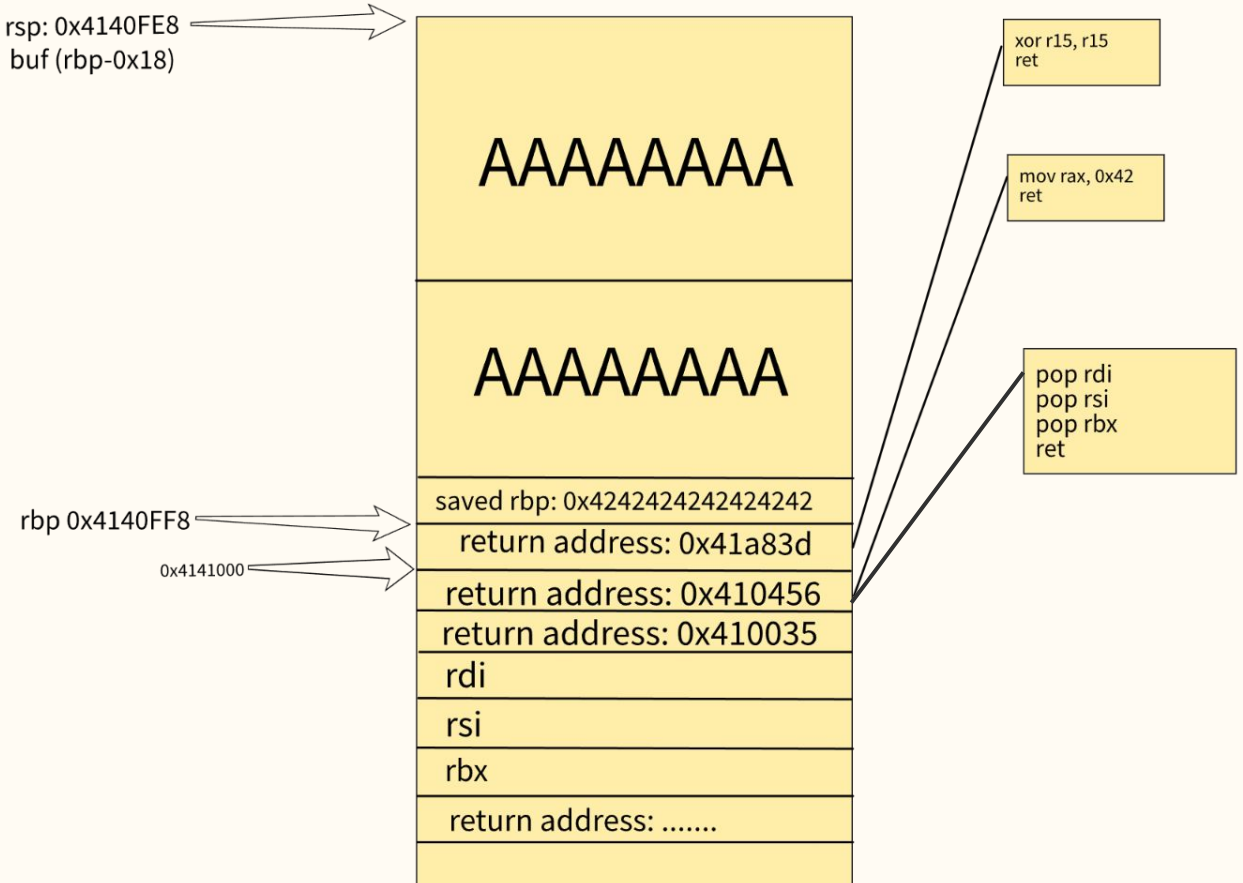
```
0x410456
   mov rax, 0x42
   ret
```

```
0x410035
   pop rdi
   pop rsi
   pop rbx
   ret
```

What is a ROP chain?

A set of gadgets that accomplishes some task. It is comprised of addresses to execute and data if there are pops, etc.

Chain in memory:



How this would look in python:

```
data = b’a’*16
data += b’c’*8 ## rbp
data += p64(0x41a83d)
data += p64(0x410456)
data += p64(0x410035)
data += p64(0xdeadbeef) #rdi
data += p64(0xd00dd00d) #rsi
data += p64(0x41414141) #rbx
data += ……
```

ret2libc

- This is ROP but while using libc functions.
- If ASLR is enabled then you need a leak or a partial overwrite.
- Functions: system; execve, mmap, etc.
- One_shot

Chain example

- mmap() - Allocate RWX memory
- Rdi - rsi
  - Pop rdi; ret
- read() - read shellcode into memory
- jmp to shellcode

Leveraging Variable Width Instructions

- Intel instructions can be from 1-15 bytes
- You do not have to start executing at the first byte.
- Depending on where you start you will get different instructions

```
0x0000000010000000 in _start ()
(gdb) x /i $pc
=> 0x10000000 <_start>: mov    rax,0xc3050f
(gdb) x /i $pc+1
   0x10000001 <_start+1>:       mov    eax,0xc3050f
(gdb) x /i $pc+2
   0x10000002 <_start+2>:       ror    BYTE PTR [rdi],0x5
(gdb) x /i $pc+3
   0x10000003 <_start+3>:       syscall
(gdb) x /i $pc+4
   0x10000004 <_start+4>:       add    eax,0xc3
(gdb) x /i $pc+5
   0x10000005 <_start+5>:       ret
(gdb)
```

Tools

- ROPGadget (python3-ropgadget)

```
hj@diocletian:~/basic-binary-exploitation/midterm/question-5$ ROPgadget --binary
question-5
Gadgets information
============================================================
0x0000000000001137 : adc byte ptr [rax], al ; add byte ptr [rax], al ; jmp 0x1020
0x0000000000001077 : add al, 0 ; add byte ptr [rax], al ; jmp 0x1020
…
0x00000000000011db : test rax, rax ; je 0x11e8 ; jmp rax

Unique gadgets found: 148
```