Binary exploitation workshop

Martin Meyers

Radboud Institute of Pwning

20240112

Whoami

- Martin Meyers
- Msc Student @ Radboud University
- One of the founders of the current iteration of the CTF team there
  - Find us on Discord: `https://discord.gg/bD8D7S5euv`
  - And our blog at: `https://pwning.nl/`

Overview

Stack

Buffer overflow

- Basics
- Return Oriented Programming
- ASLR
- Return to libc

Extras

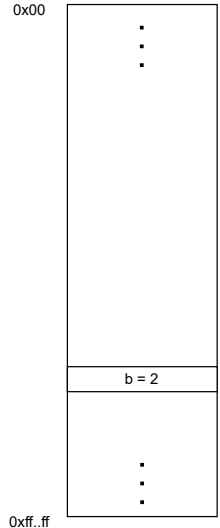
- Format string attacks
- Some Linux specifics

Stack

- A part of memory that local data is typically stored in
- Grows up

```
int a = 1;

int main() {
        int b = 2;
}
```

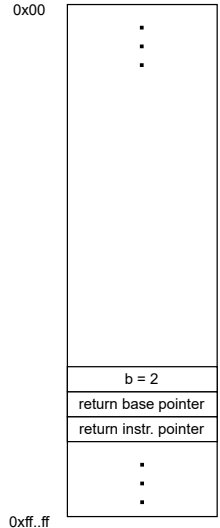


Stack

- Place for the stackframes

```
int main() {
        int b = 2;
}
```

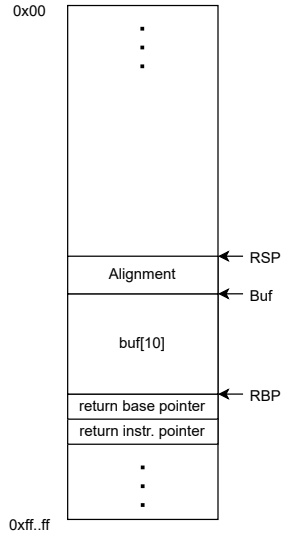
```
push    rbp
mov     rbp, rsp
mov     DWORD PTR [rbp-0x4], 0x2
mov     eax, 0x0
pop     rbp
ret
```



Stack

```
void f() {  char buf[10];
            gets(buf);  }
```

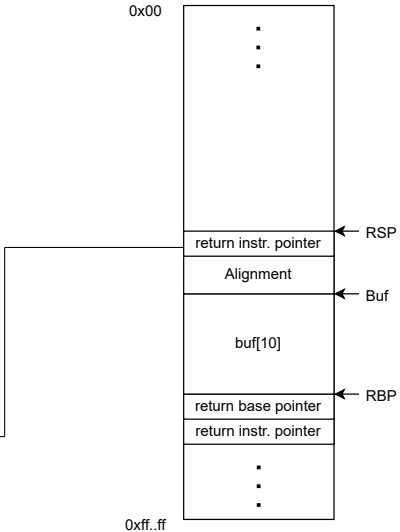
```
push    rbp
mov     rbp,rsp
sub     rsp,0x10
lea     rax,[rbp-0xa]
mov     rdi,rax
mov     eax,0x0
call    1040 <gets@plt>
nop
leave
ret
```



Stack

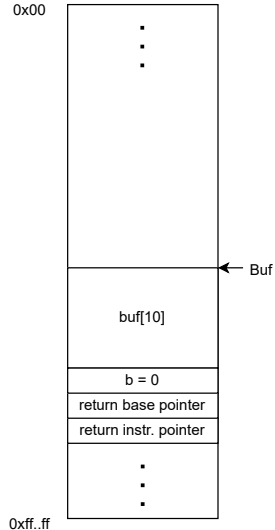
```
void f() {  char buf[10];
            gets(buf);  }
```

```
push    rbp
mov     rbp,rsp
sub     rsp,0x10
lea     rax,[rbp-0xa]
mov     rdi,rax
mov     eax,0x0
call    1040 <gets@plt>
nop
leave
ret
```



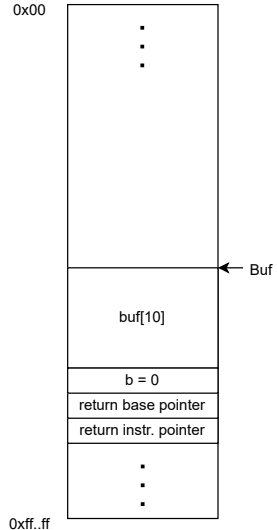
Buffer overflow

```
void f() {
        int b = 0;
        char buf[10];
        gets(buf);
}
```



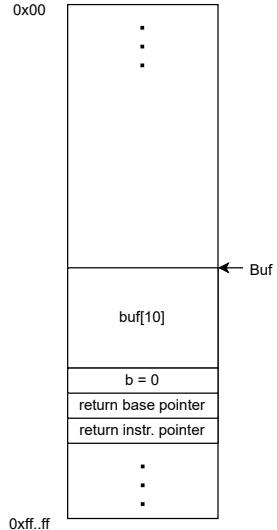
Challenge 0

Demo time!



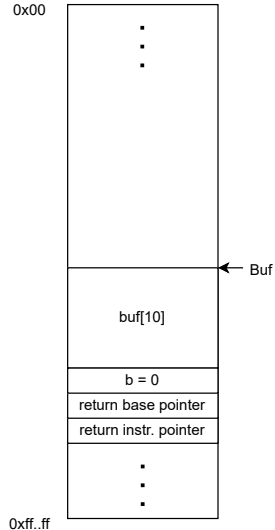
Challenge 1

Your turn!



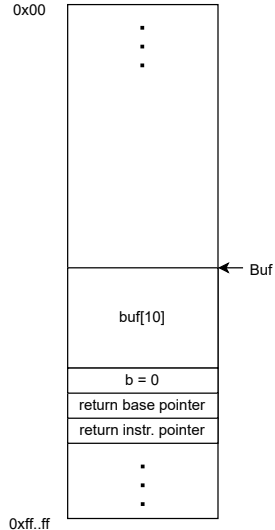
Return Pointer

- Overwrite the Return Instruction Pointer



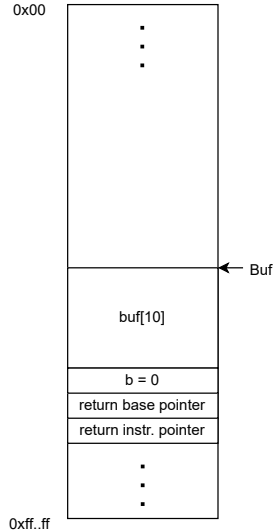
Return Pointer

- Overwrite the Return Instruction Pointer
- The return will jump where you want!



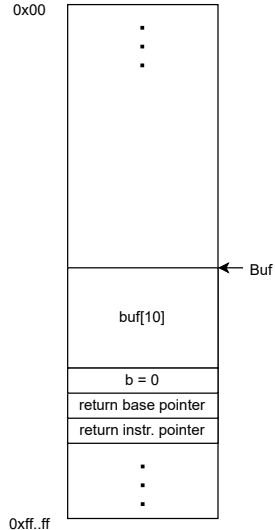
Shellcode

- Write your own code and jump to it!



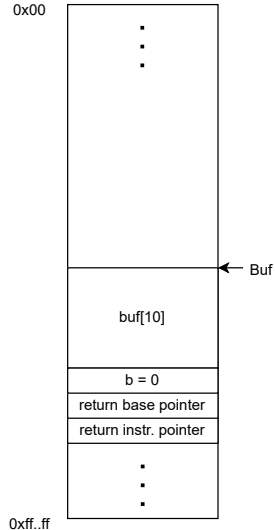
Shellcode

- Write your own code and jump to it!
- Well, often no...
  - NX
  - ASLR



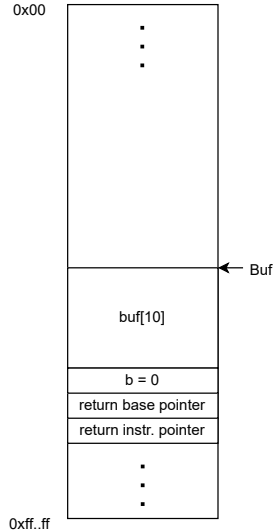
Return Oriented Programming

- Just use existing code
- No issues with NX



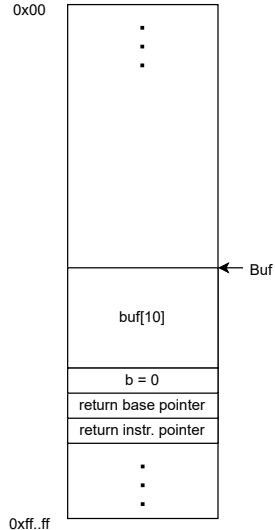
Intermezzo

How do you put a pointer to code there?



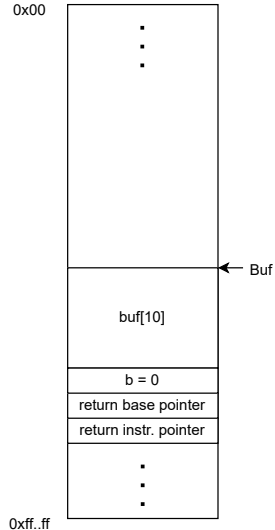
Challenge 2

Good luck!



ROP chaining

- ROP gadgets
- Keep returning



x86 Calling Convention

Function arguments are in:

1. RDI
2. RSI
3. RDX
4. RCX
5. R8
6. R9
7. and further on the stack

- Return value in RAX

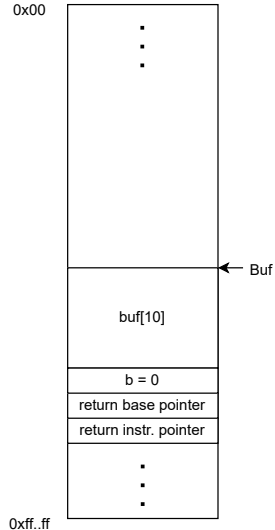
Linux x86-64 specific (and simplified)!

Why is this important?

```
int winner(char value) {
        if (value == 'W')
                printf("You-win!\n");
        else
                printf("Wrong-value!\n");
}
```

Challenge 3

- First argument in RDI
- Good luck!



ASLR

- Address
- Space
- Layout
- Randomization

ASLR

- Address
- Space
- Layout
- Randomization

Position Independent Executable (PIE)

ASLR

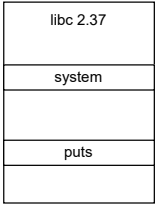
- GDB disables it when you run through there
- Can see mapping in GDB using:
  - (pwndbg) vmmap
  - (otherwise) info proc mappings

How to defeat ASLR

- Leak address
- Calculate base address
- Use offset

To find the offsets, use:

- Pwntools ELF symbols
- Any disassembler

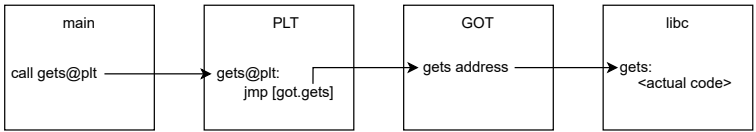


Challenge 4

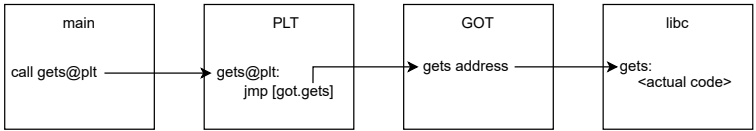
- Good luck!
- We advice to use pwntools - though it is not strictly necessary (yet).
  A template script can be found at `https://pwning.nl/posts/how2pwn/`

PLT/GOT

- Procedure Linkage Table
- Global Offset Table

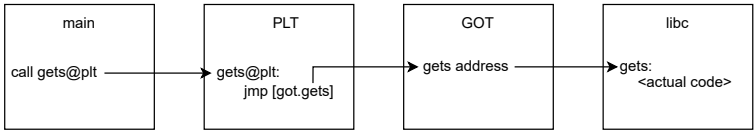


Leaking an address from GOT



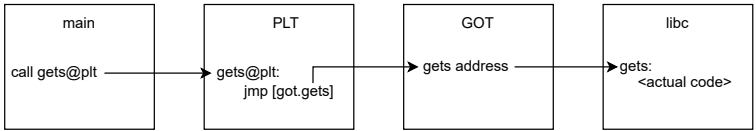
- Have no PIE
- Be creative

Leaking an address from GOT



- Have no PIE
- Be creative
- Use puts or %s format string
- Generally two-stage payload
- Sometimes try a different symbol

Challenge 5



Good luck with the challenge!

Challenge 5

And that was a return to libc challenge!
Any questions?

Bonus challenge

Challenge 6 is a bonus challlenge!

Format string attacks

- The %n format specifier can write data
- %40x will print 40 characters
- %10$n will use the 10th argument
- Does not work on Windows by default

Some Linux specifics

- man shows dangerous functions
- File Descriptors are kept open through exec calls
- There is a lot of information in the /proc filesystem
- You can search an entire disk through the /dev files

The end

And that is the real end!