Course Review

Yajin Zhou (http://yajin.org)

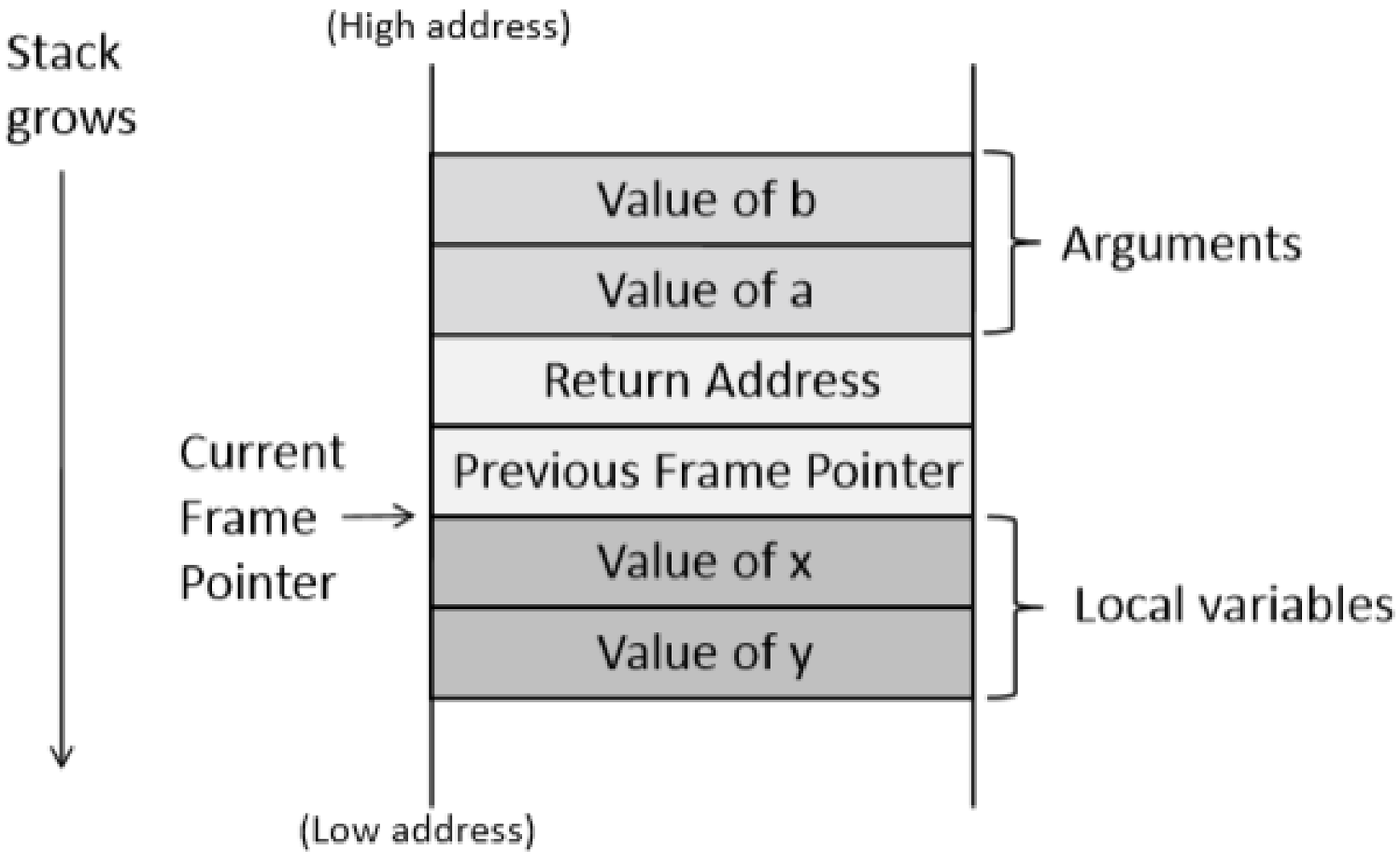
Zhejiang University

Credits: SEEDLab
http://www.cis.syr.edu/~wedu/seed/

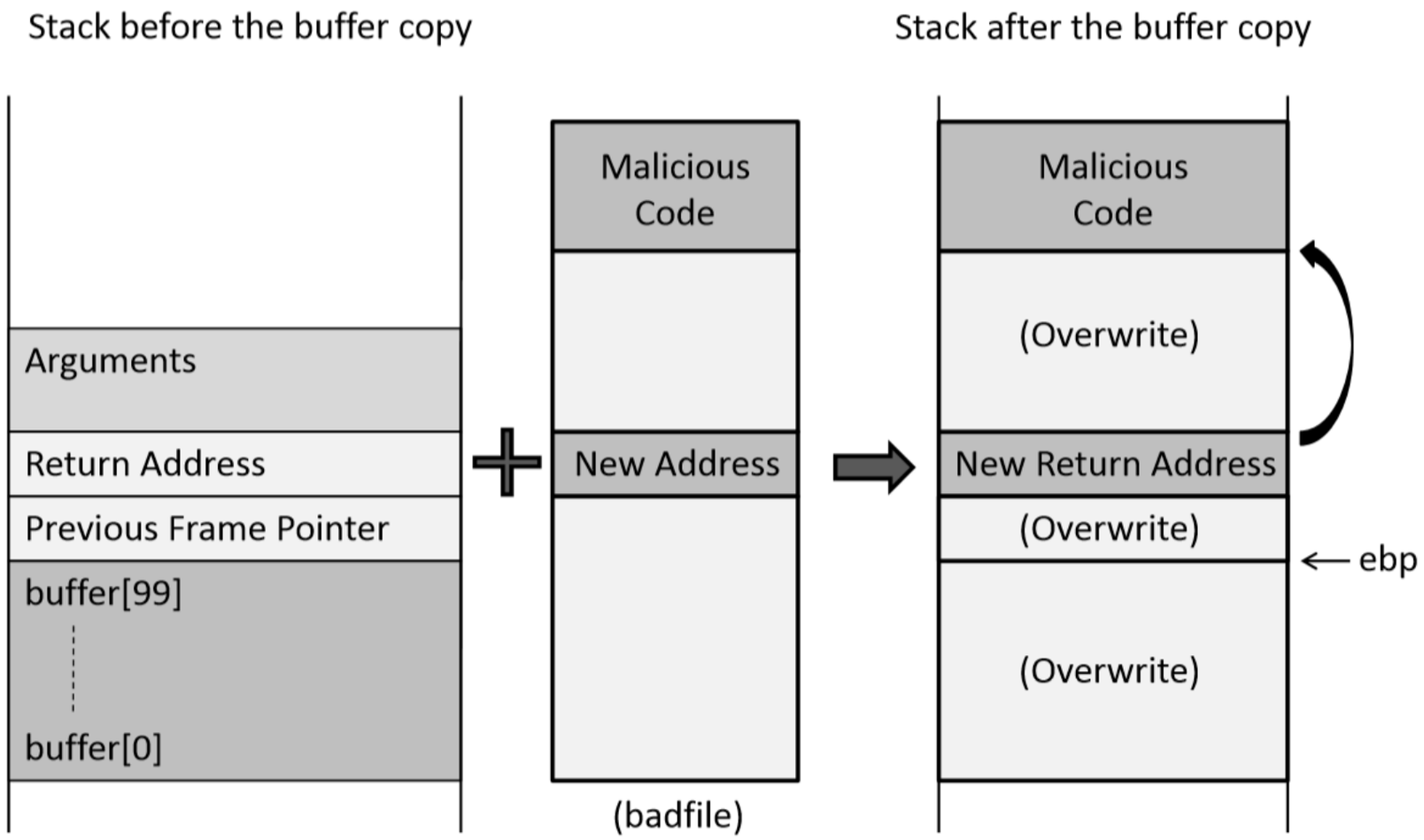
What we have learnt

- Buffer overflow
- Return2libc (with/without ASLR)
- Heart bleeding
- Set-UID, Environment variables
- Format String Vulnerability
- Race Condition/Dirty COW
- Smart Contract Security

Stack Layout

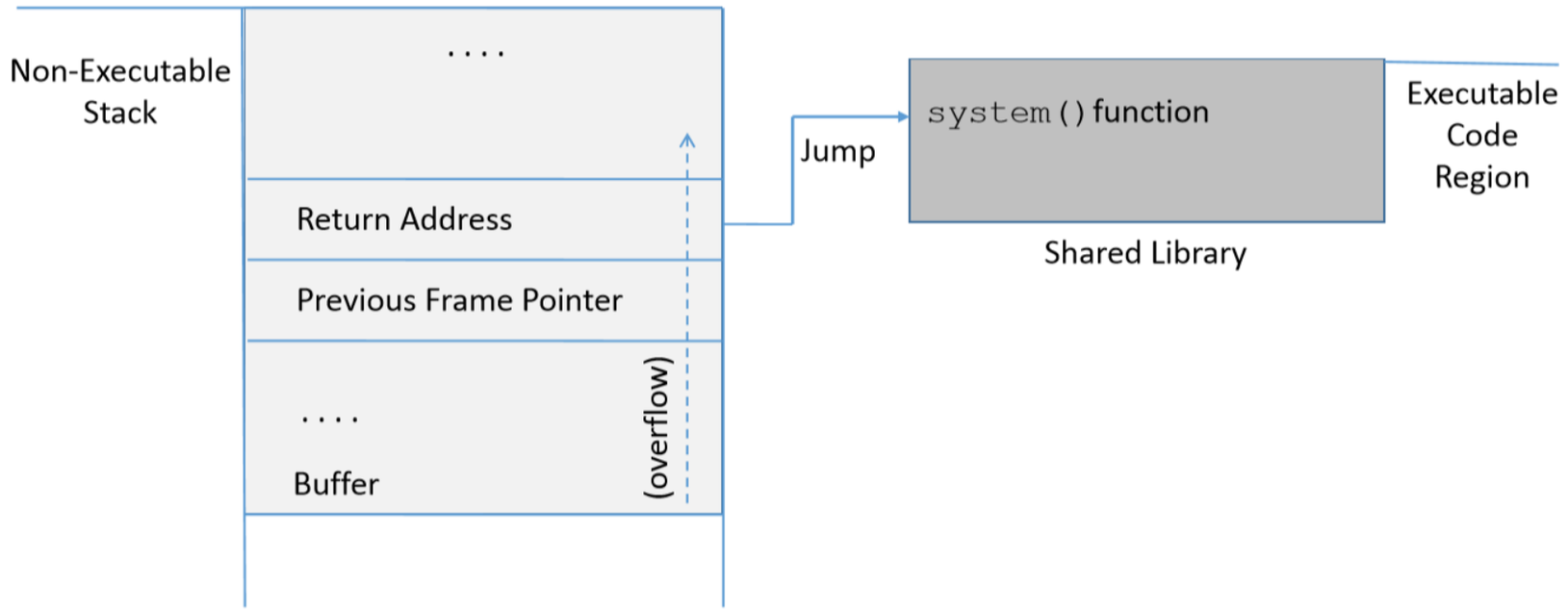


How to Exploit



The Idea of Return-to-libc

- In fact, the process' memory space has lots of code that could be **abused**



How to attack

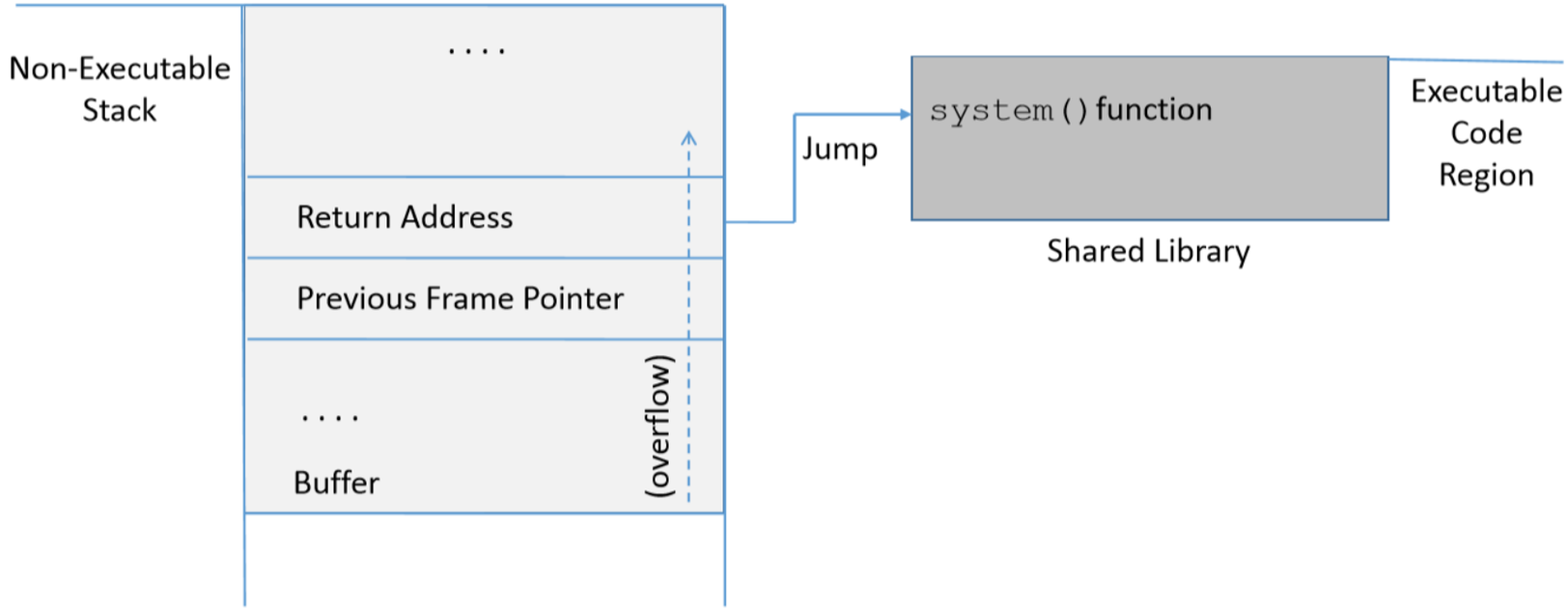
1. **任务 A：找到 system() 的地址**。我们需要找到 system() 函数在内存中的位置。我们将修改函数的返回地址为该地址，这样函数返回时候程序就会跳转到 system()。

2. **任务 B：找到字符串 “/bin/sh” 的地址**。为使 system() 函数运行一个命令，命令的名字需要预先在内存中存在，并且能够获取它的地址。

3. **任务 C：system() 的参数**。获取字符串 “/bin/sh” 地址之后，我们需要将地址传给 system() 函数。这意味着需要把地址放在栈中，因为 system() 从栈中获取参数。难点在于我们弄清应该将地址具体放置在哪个位置。

The Idea of Return-to-libc

- In fact, the process' memory space has lots of code that could be **abused**



What if we have ASLR enabled?

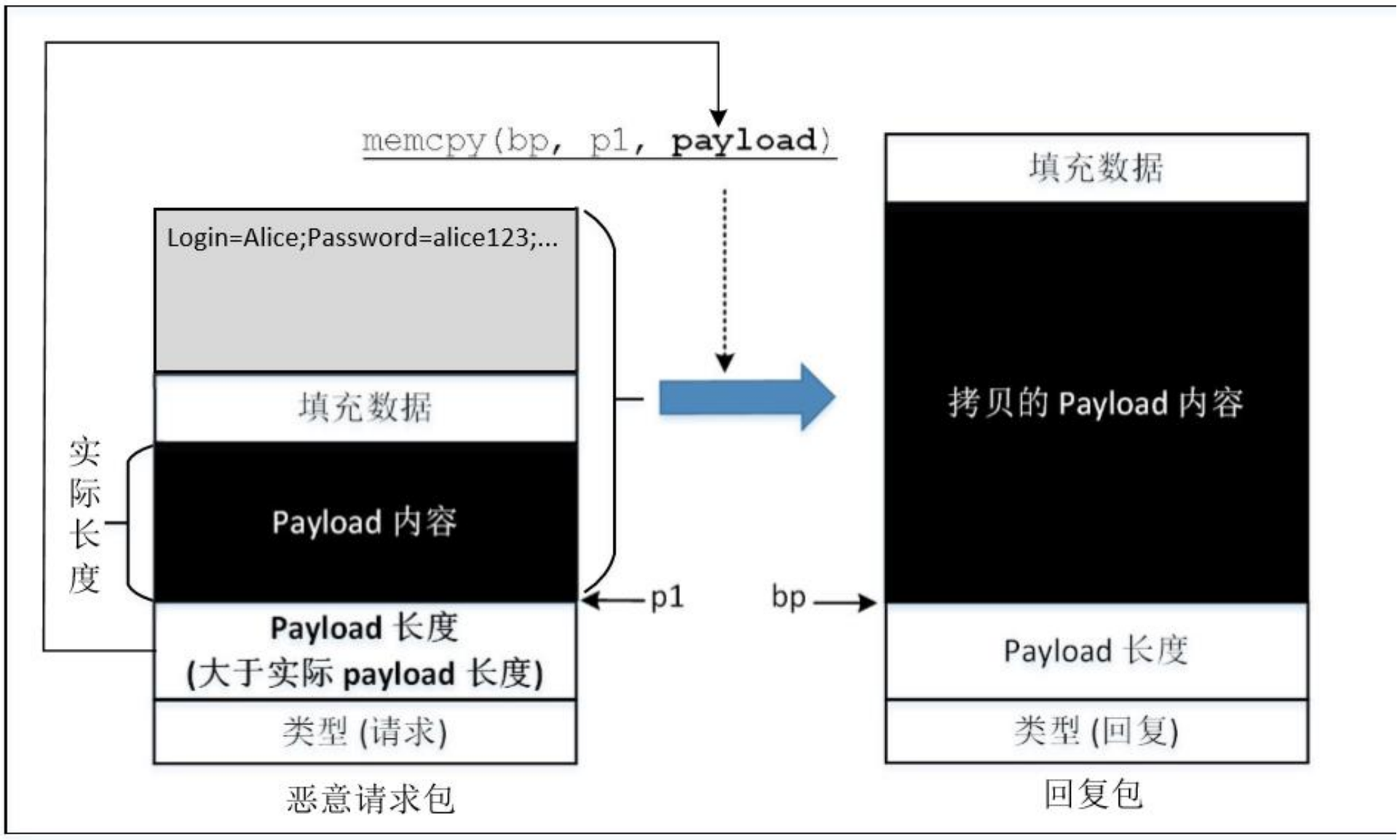
Leak Libc Base

- Dynamic linking

```
08048380 <puts@plt>:
 8048380:    ff 25 14 a0 04 08          jmp    *0x804a014
 8048386:    68 10 00 00 00             push   $0x10
804838b:    e9 c0 ff ff ff             jmp    8048350 <_init+0x24>
```

```
gef➤  x/xw 0x804a014
0x804a014 <puts@got.plt>:    0x08048386
```

Heartbleeding: How To Exploit



Shellshock

- CVE-2014-6271, exists since 1989
- The shell will execute command after }

```
$ foo='() { echo "hello world"; }; echo "extra";'
$ echo $foo
() { echo "hello world"; }; echo "extra";
$ export foo
$ bash_shellshock     ← 运行有漏洞的 bash 版本
extra                 ← 额外的命令被执行了!
(child):$ echo $foo

(child):$ declare -f foo
foo ()
```

```
{
    echo "hello world"
}
```

A vulnerable program

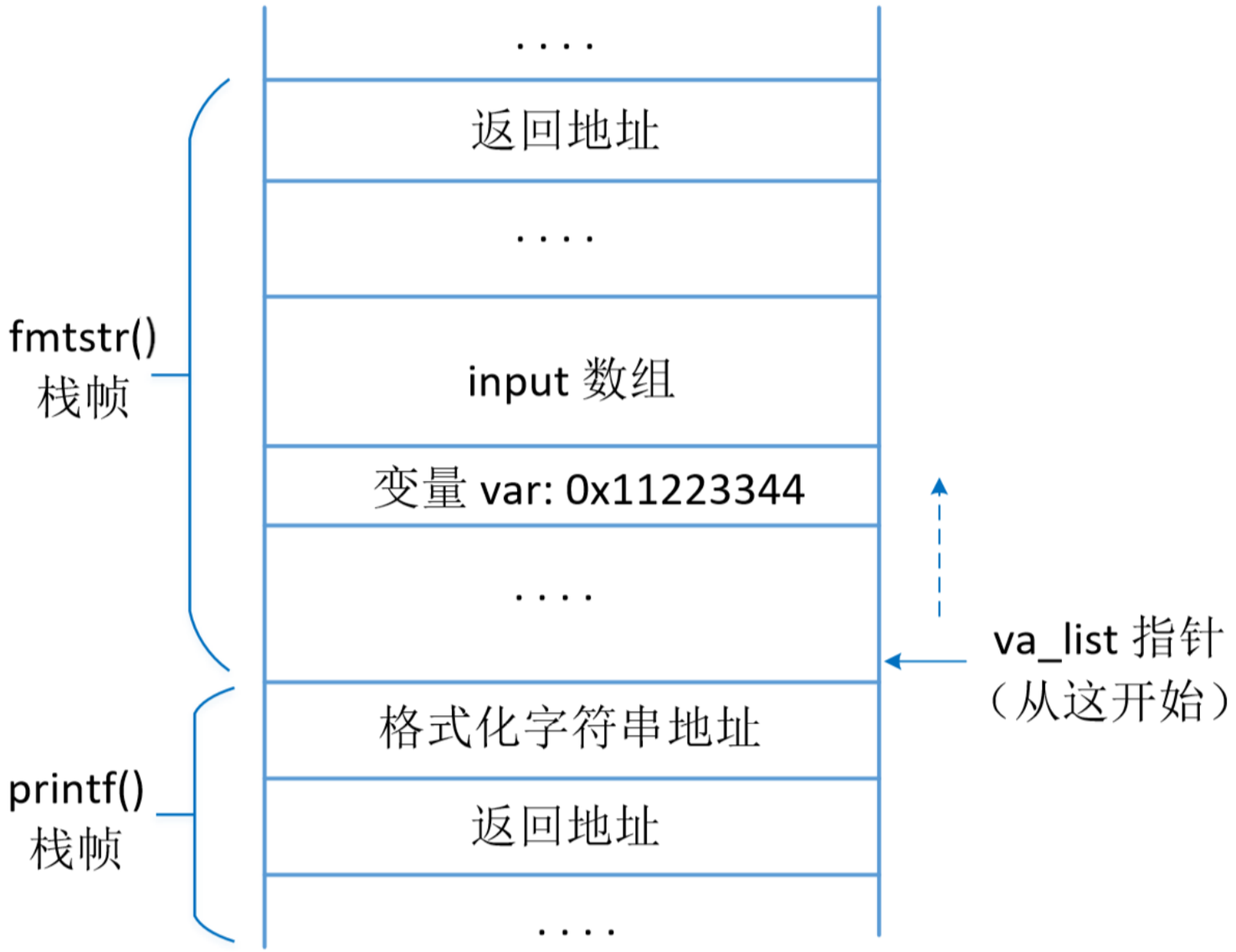


图 6.4: 漏洞程序栈帧的布局

write()

步骤 A: 给映射的内存做一份拷贝

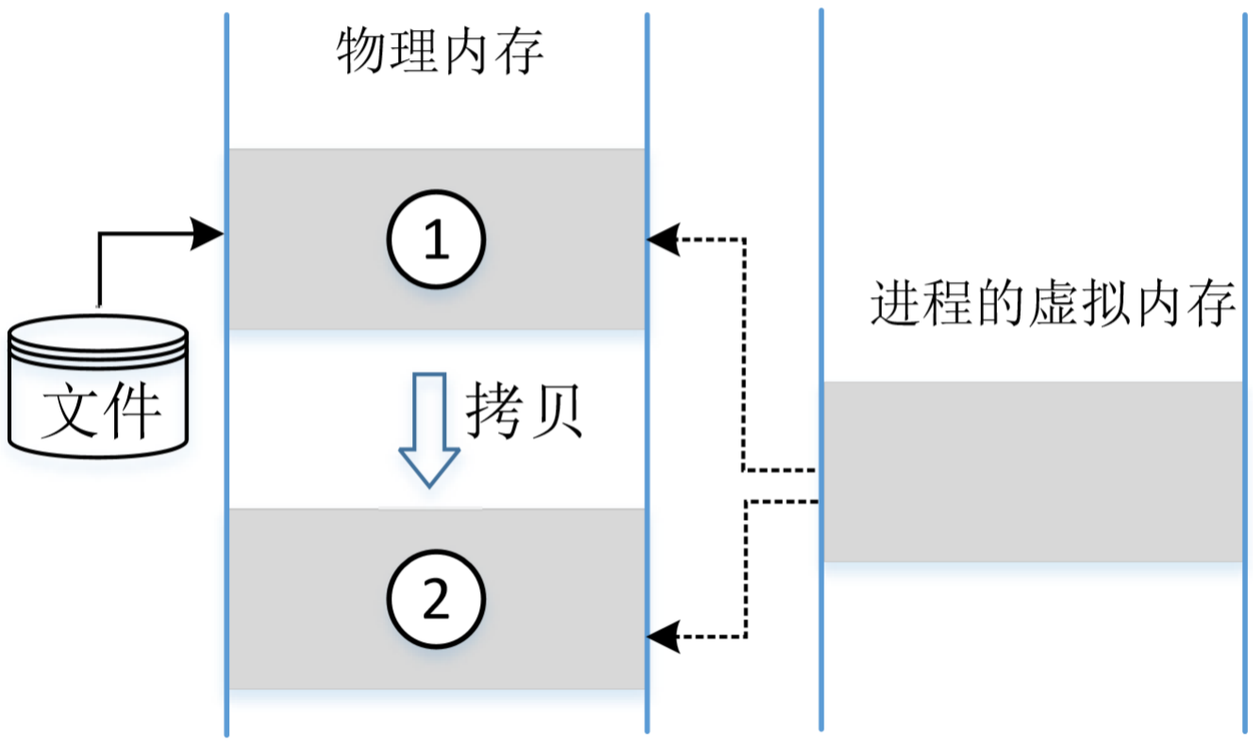
步骤 B: 修改页表，使虚拟内存指向 ②

步骤 C: 往内存里写

madvice():

用 MADV_DONTNEED

修改页表，使虚拟内存指向 ①



Thanks!