第一次二进制培训

Vidar-Team

VIDAR TEAM

为什么要学二进制?

VIDAR TEAM

二进制有哪些方向

- 逆向工程
- PWN

VIDAR TEAM

什么是逆向工程？

一种探究应用程序内部组成结构及工作原理的技术

VIDAR TEAM

- 轻松窥探程序内部结构、掌握工作原理
- 在程序的开发与测试阶段发现BUG和漏洞
- 并直接修改程序文件或内存解决这些隐含的问题
- 为程序添加新功能，使程序更强大

这就像是一种魔法，魅力无限

VIDAR TEAM

什么是PWN?

VIDAR TEAM

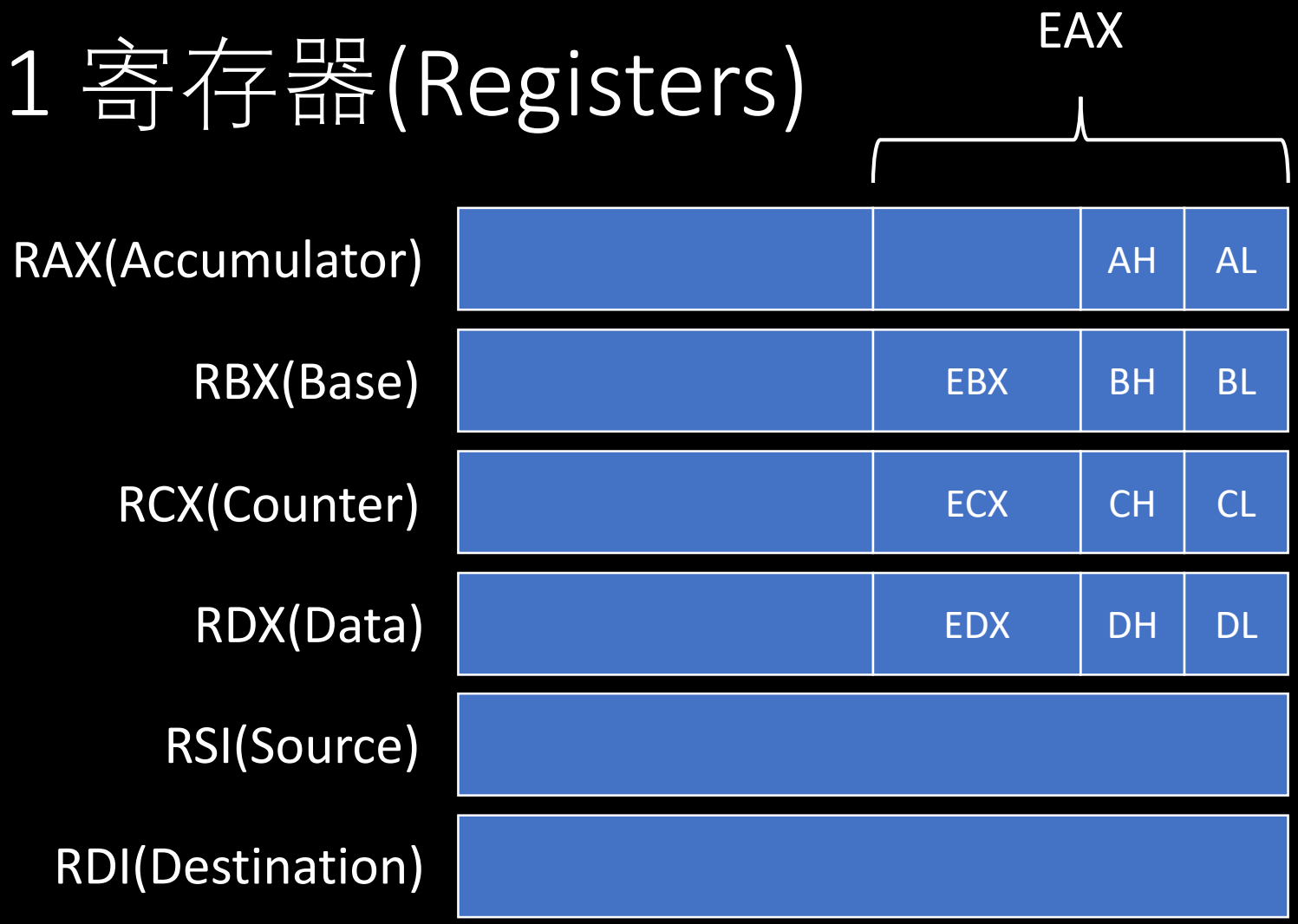
控制一个程序的流程，触发攻击

VIDAR TEAM

1. 反汇编

VIDAR TEAM

1.1 寄存器(Registers)



== 8 bit

VIDAR TEAM

1.1 寄存器(Registers)

RSP(Stack Pointer)

RBP(Stack Base Pointer)

RIP(Instruction Pointer)

 == 8 bit

VIDAR TEAM

1.1 汇编语言

Add a, b: a += b
Sub a, b: a -= b
Jmp a: 跳转到a地址
Call a: 调用a函数
Cmp a, b: a与b进行比较
Mov a, b: 把b的值赋值给a
Ret: 返回到栈顶指向的地址
Nop: no operation
Push a: 将a入栈
Pop a: 出栈并存入a中

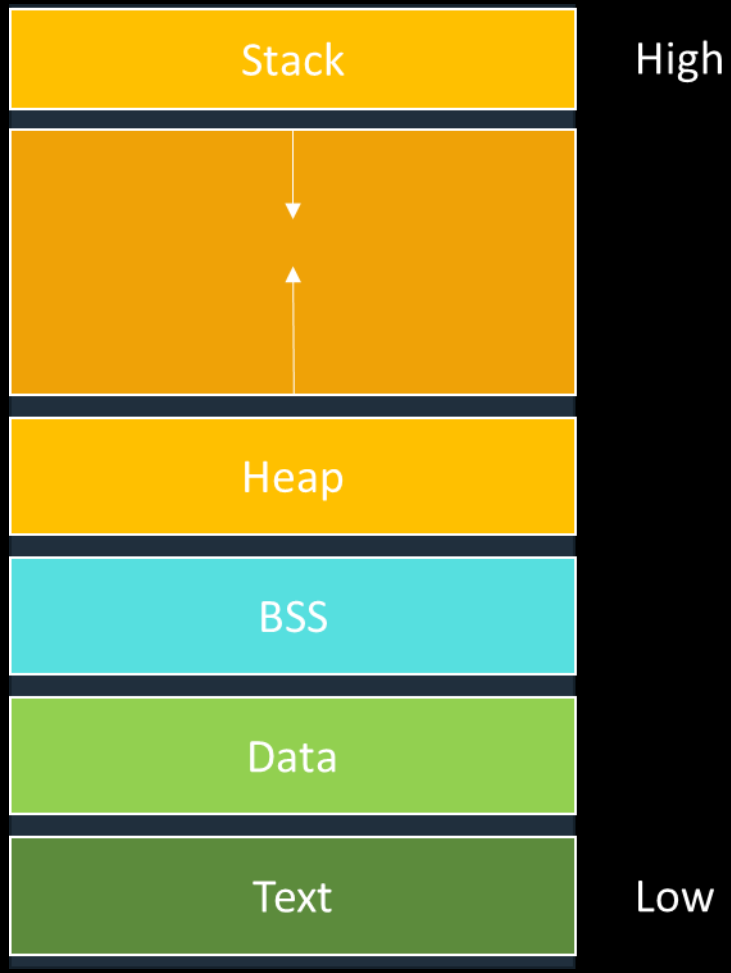
VIDAR TEAM

1.1 汇编语言 来看一个demo

VIDAR TEAM

2. Program Structure

2.1



VIDAR TEAM

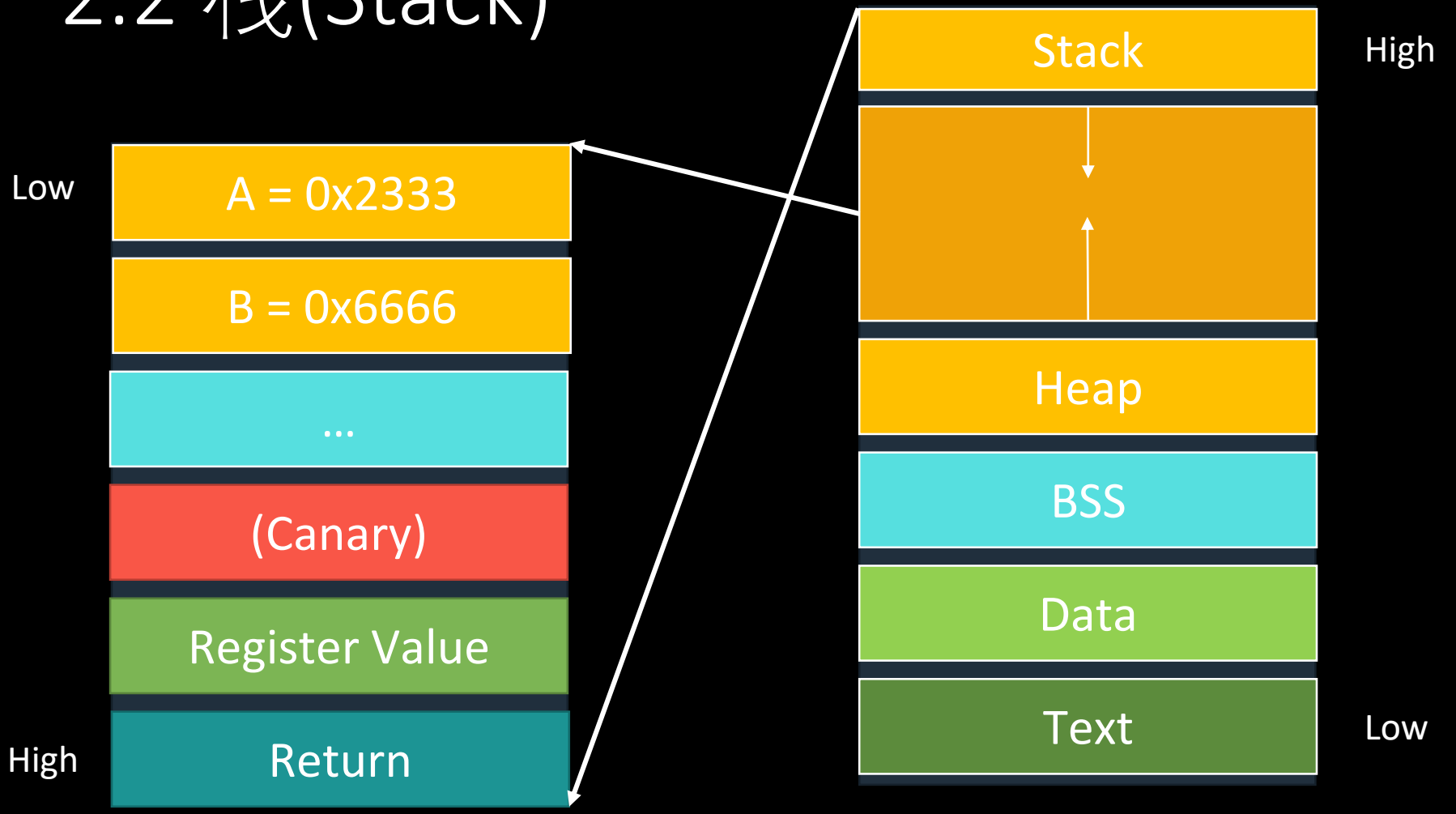
2.1

```c
#include <stdio.h>

int c = 0x1234;
int d;

void myFunc()
{
    int a = 0x2333;
    int b = 0x6666;
    printf("%d, %d", a, b);
}

int main()
{
    c = 0x4321;
    d = 0x1551;
    myFunc();
}
```

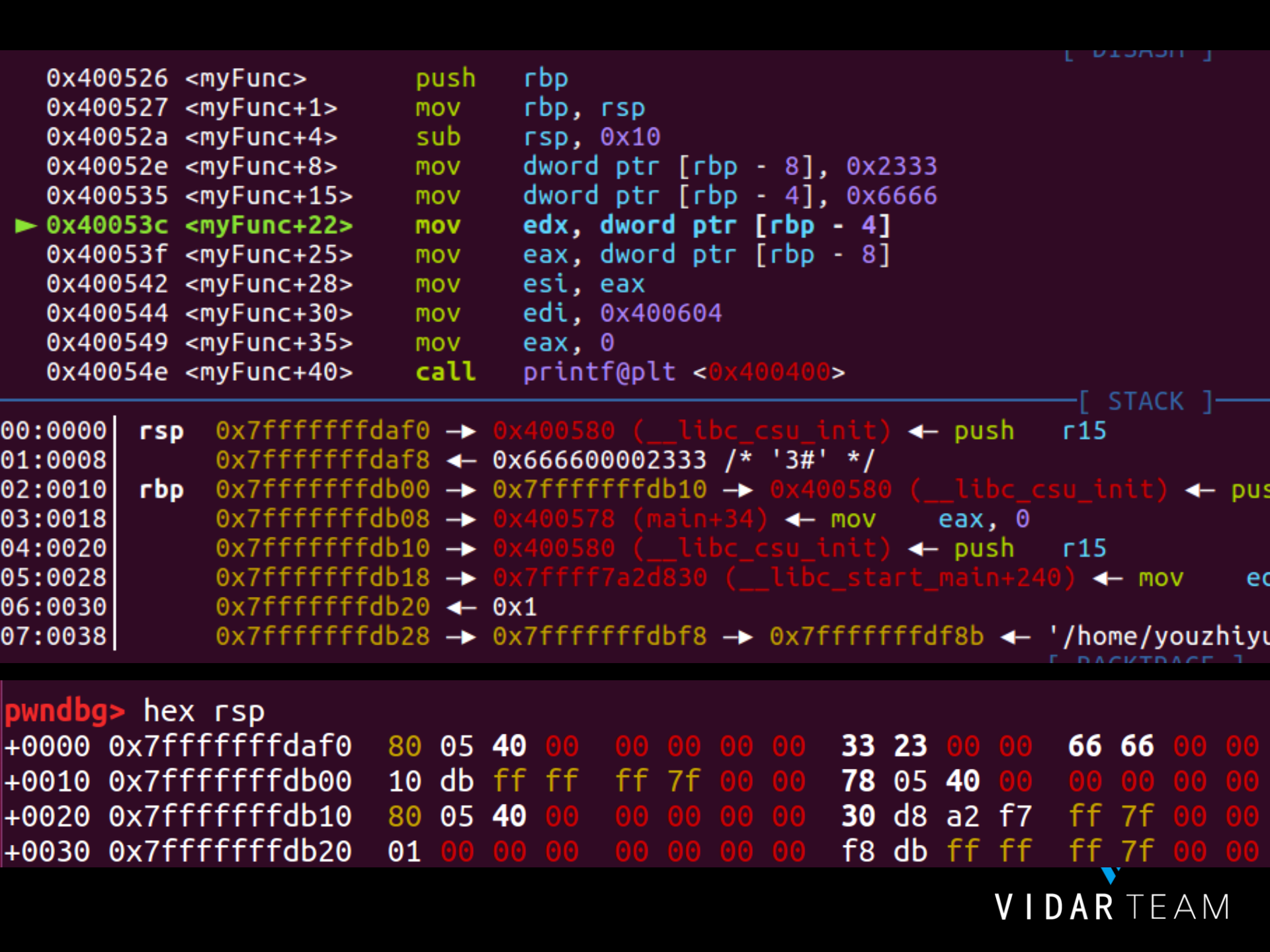
VIDAR TEAM

2.1 我们可以用IDA看一看

2.2 栈(Stack)



VIDAR TEAM


0x400526 <myFunc> push rbp
0x400527 <myFunc+1> mov rbp, rsp
0x40052a <myFunc+4> sub rsp, 0x10
0x40052e <myFunc+8> mov dword ptr [rbp - 8], 0x2333
0x400535 <myFunc+15> mov dword ptr [rbp - 4], 0x6666
0x40053c <myFunc+22> mov edx, dword ptr [rbp - 4]
0x40053f <myFunc+25> mov eax, dword ptr [rbp - 8]
0x400542 <myFunc+28> mov esi, eax
0x400544 <myFunc+30> mov edi, 0x400604
0x400549 <myFunc+35> mov eax, 0
0x40054e <myFunc+40> call printf@plt <0x400400>
[STACK]
00:0000 rsp 0x7fffffffdaf0 → 0x400580 (__libc_csu_init) ← push r15
01:0008 0x7fffffffdaf8 ← 0x666600002333 /* '3#' */
02:0010 rbp 0x7fffffffdb00 → 0x7fffffffdb10 → 0x400580 (__libc_csu_init) ← push
03:0018 0x7fffffffdb08 → 0x400578 (main+34) ← mov eax, 0
04:0020 0x7fffffffdb10 → 0x400580 (__libc_csu_init) ← push r15
05:0028 0x7fffffffdb18 → 0x7ffff7a2d830 (__libc_start_main+240) ← mov
06:0030 0x7fffffffdb20 ← 0x1
07:0038 0x7fffffffdb28 → 0x7fffffffdbf8 → 0x7fffffffdf8b ← '/home/youzhiyu
pwndbg> hex rsp
+0000 0x7fffffffdaf0 80 05 40 00 00 00 00 00 33 23 00 00 66 66 00 00
+0010 0x7fffffffdb00 10 db ff ff ff 7f 00 00 78 05 40 00 00 00 00 00
+0020 0x7fffffffdb10 80 05 40 00 00 00 00 00 30 d8 a2 f7 ff 7f 00 00
+0030 0x7fffffffdb20 01 00 00 00 00 00 00 00 f8 db ff ff ff 7f 00 00

VIDAR TEAM

2.2 Stack – Return address

- Call function前，将 return address 存进 stack
- Return时，回到 stack 中所存的地址

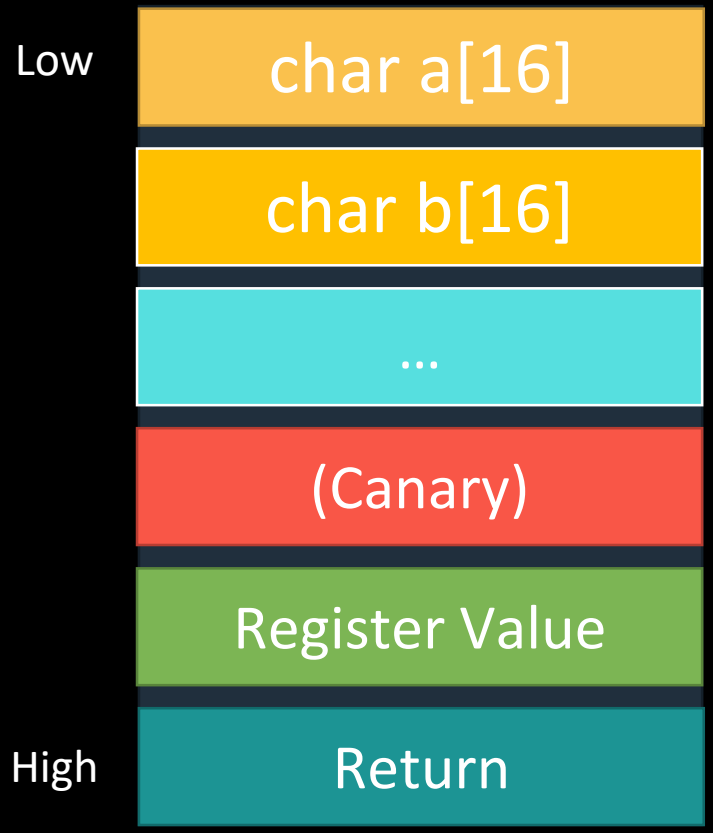
VIDAR TEAM

2.2 栈溢出

```
1   #include <stdio.h>
2
3   int main()
4   {
5       char a[16];
6       char b[16] = "hello\n";
7       gets(a);
8       printf("%s", b);
9   }
```

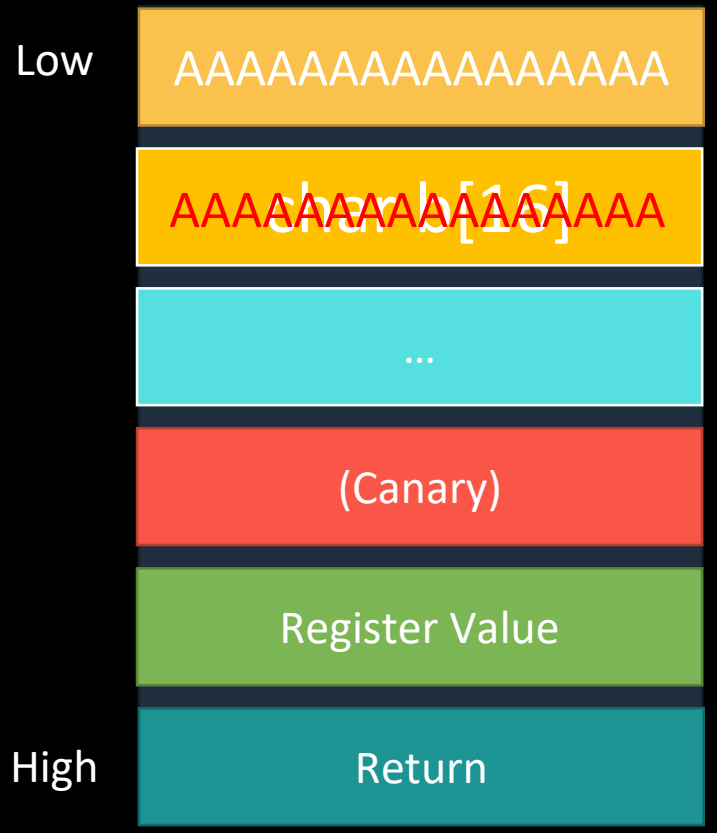
VIDAR TEAM

2.2 栈溢出



VIDAR TEAM

2.2 栈溢出



VIDAR TEAM

2.2 我们可以来动手调试一下

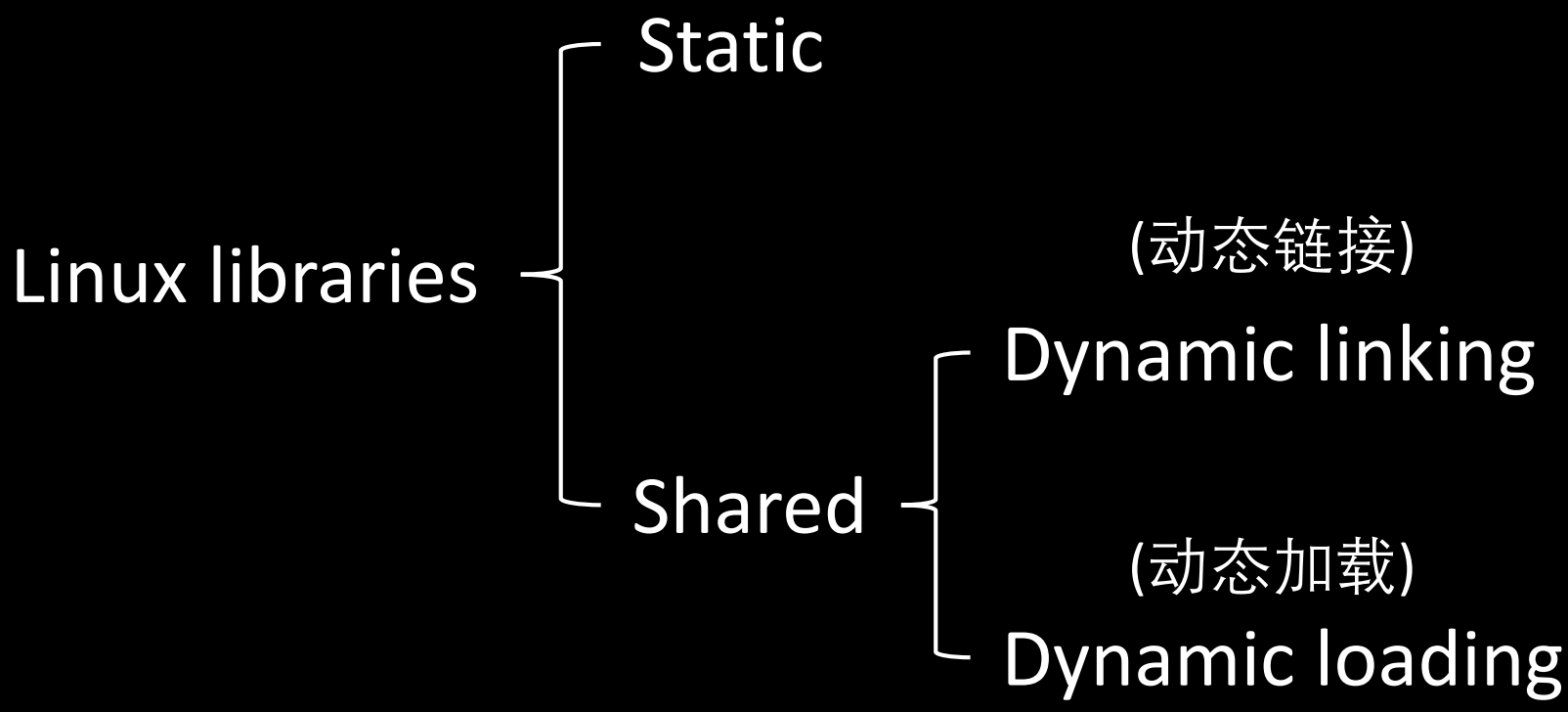
VIDAR TEAM

2.3 Security Options

```
root@kali:~/ctf/PWN# checksec test
[*] '/root/ctf/PWN/test'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
root@kali:~/ctf/PWN#
```

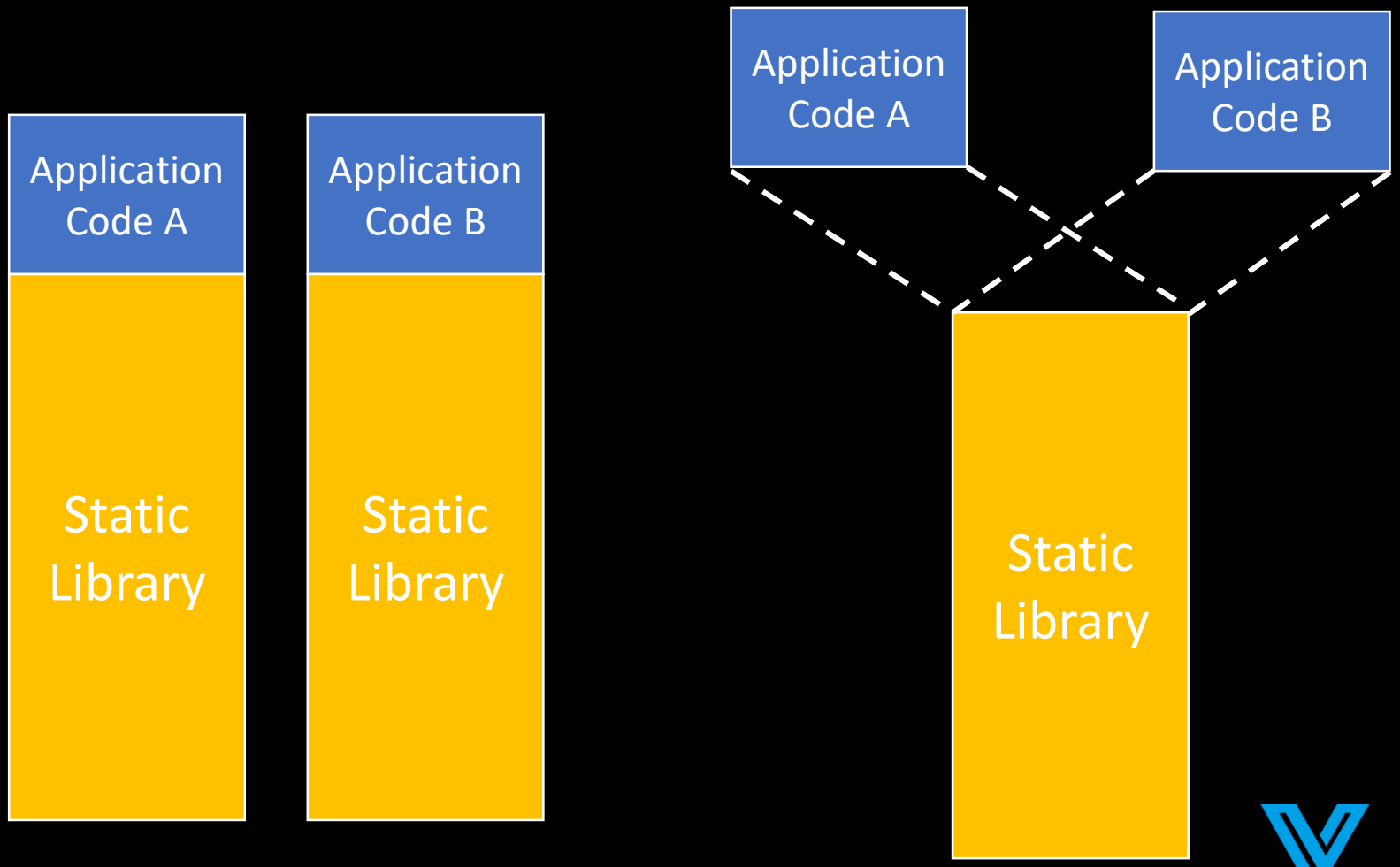
VIDAR TEAM

2.4 Linux libraries



VIDAR TEAM

2.4 Dynamic linking



VIDAR TEAM

2.4 Libc – C standard library

Ubuntu 16.04 -- Libc 2.23
Ubuntu 18.04 -- Libc 2.27+

VIDAR TEAM

2.5 Lazy Binding

在第一次 call library 函数时，才会去寻找函数真正的位置进行binding

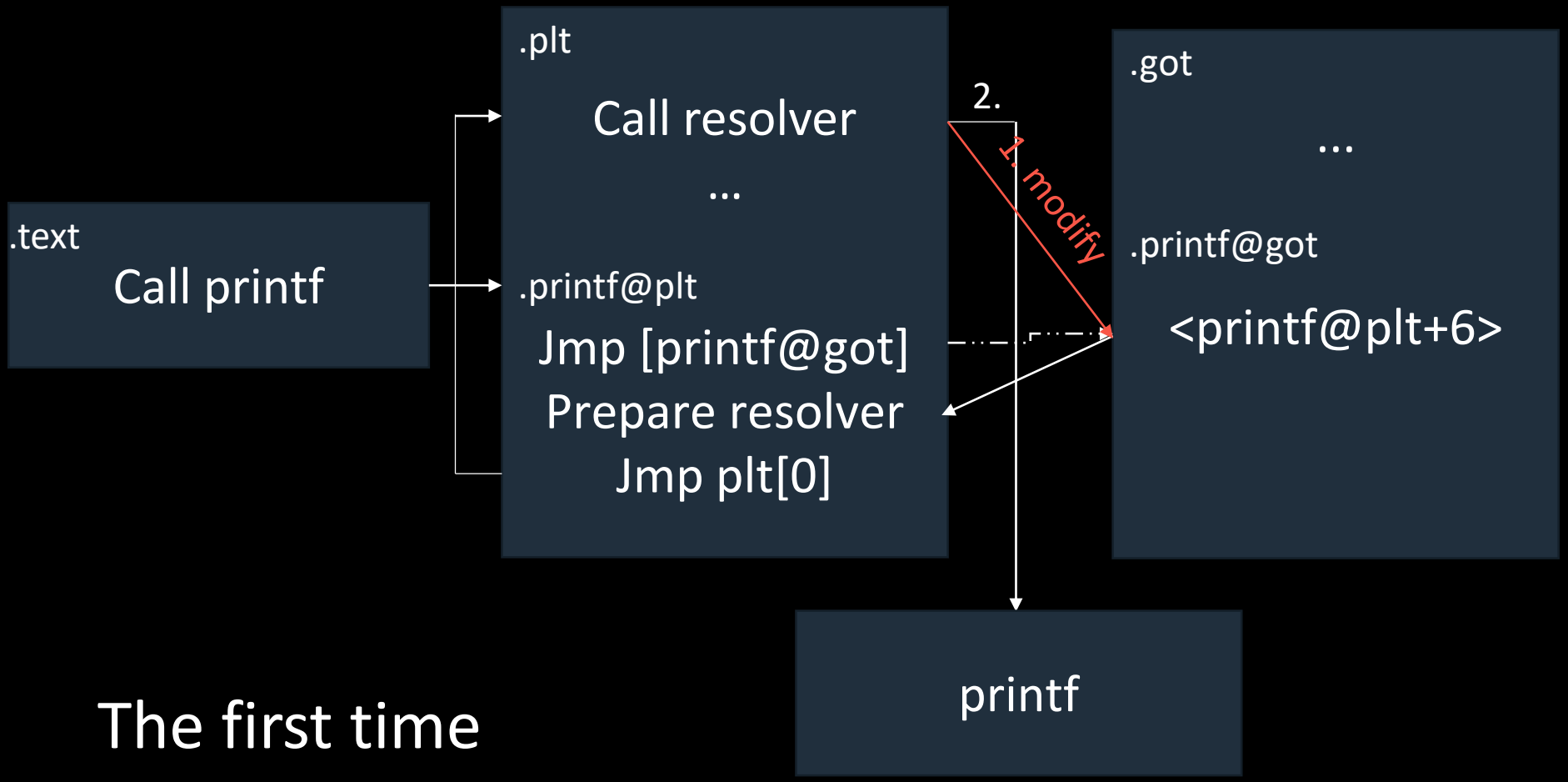
VIDAR TEAM

2.6 GOT & PLT

- GOT (Global Offset Table, 全局偏移表)
- PLT (Procedure Linkage Table, 过程连接表)

VIDAR TEAM

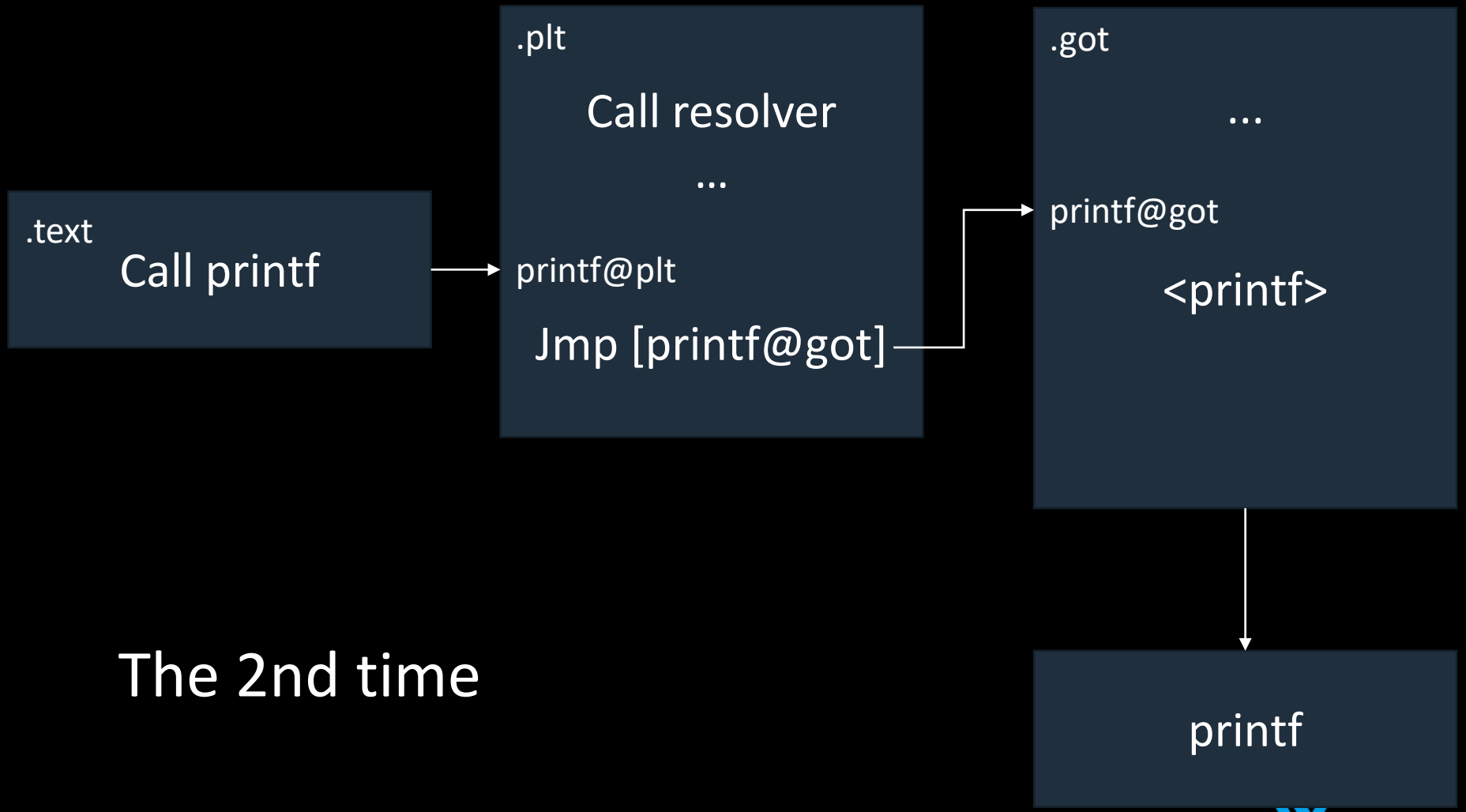
2.6 GOT & PLT



The first time

VIDAR TEAM

2.6 GOT & PLT



The 2nd time

2.7 CALL

```
.text
                                ...
   0x000011d3 e868feffff     call printf
0x000011d8 b800000000      mov eax, 0
                                ...
```

==

Push
0x000011d8

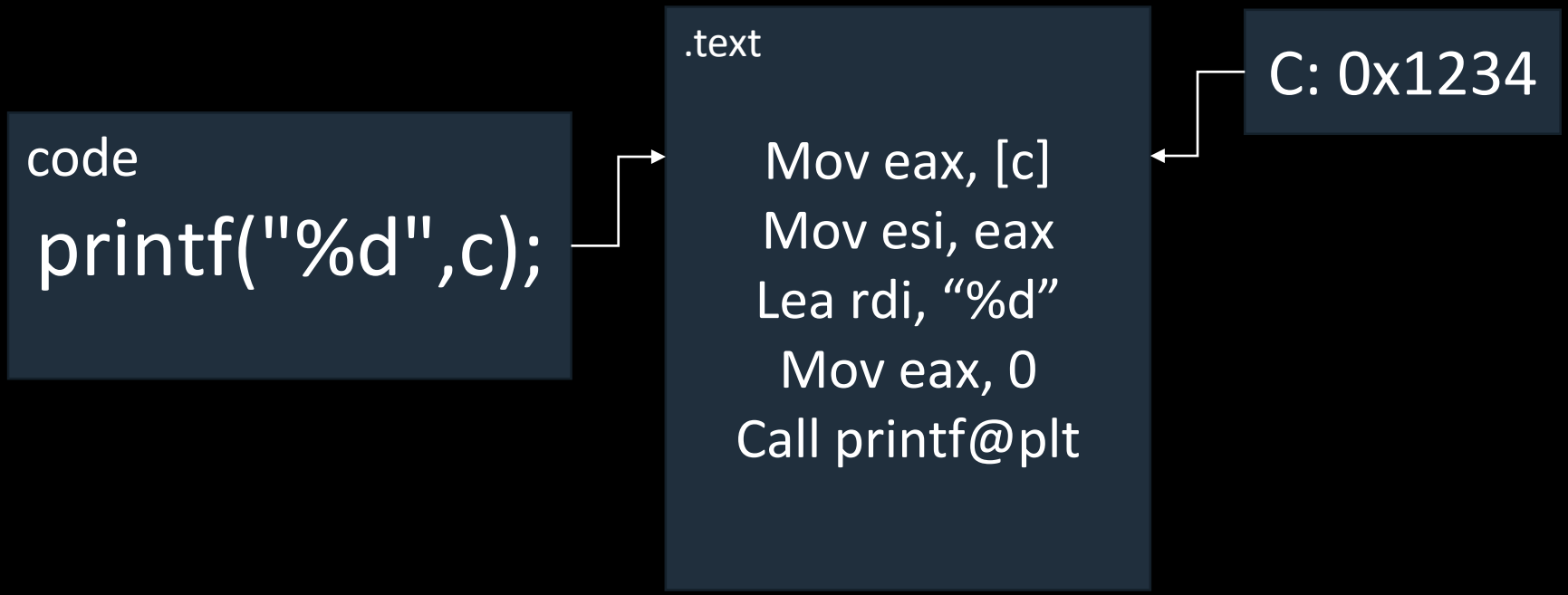
+

Jmp printf

VIDAR TEAM

2.7 CALL FUNC (x64)

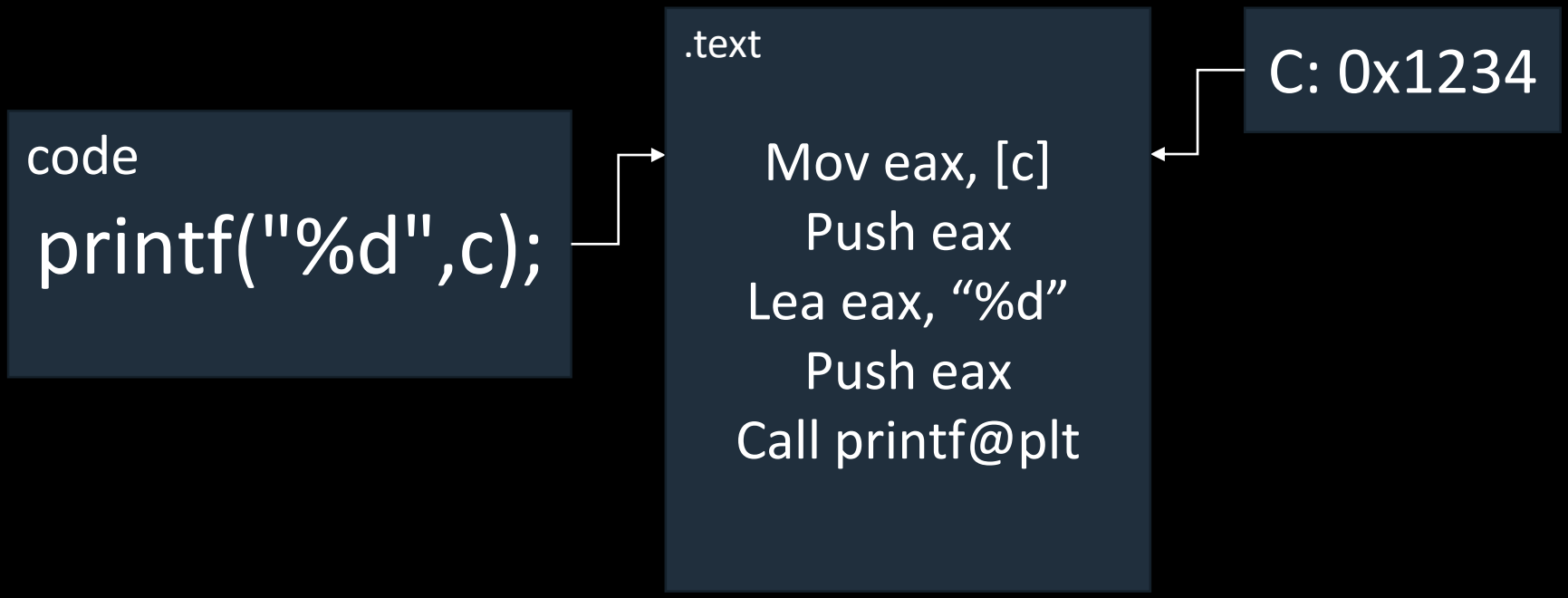
rdi	rsi	rdx	rcx	r8	r9	stack



VIDAR TEAM

2.8 CALL FUNC (x86)

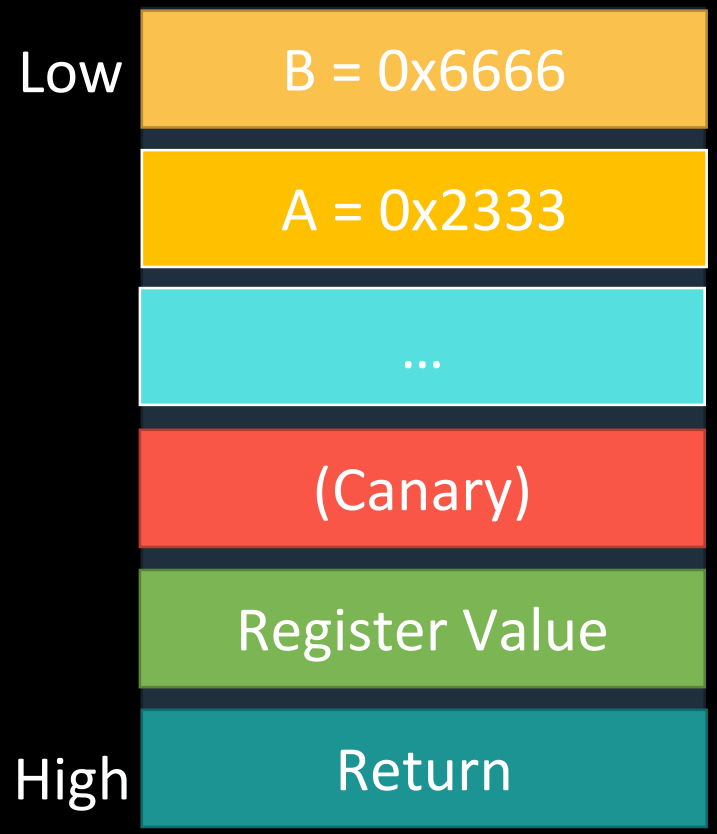
stack



VIDAR TEAM

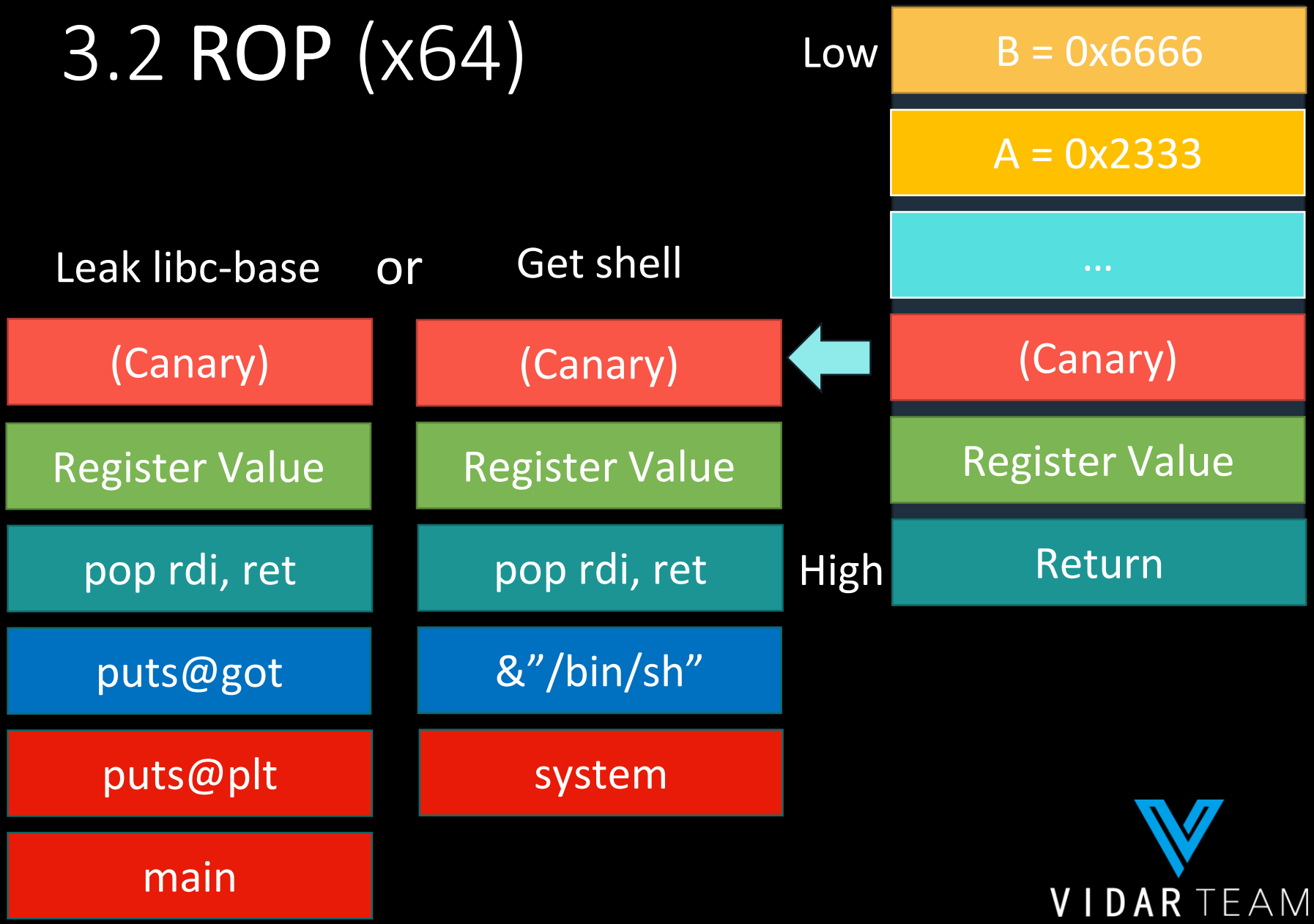
3.1 ROP (Return Oriented Programming)

栈缓冲区溢出的基础上，利用程序中已有的小片段 (gadgets) 来改变某些寄存器或者变量的值，从而控制程序的执行流程



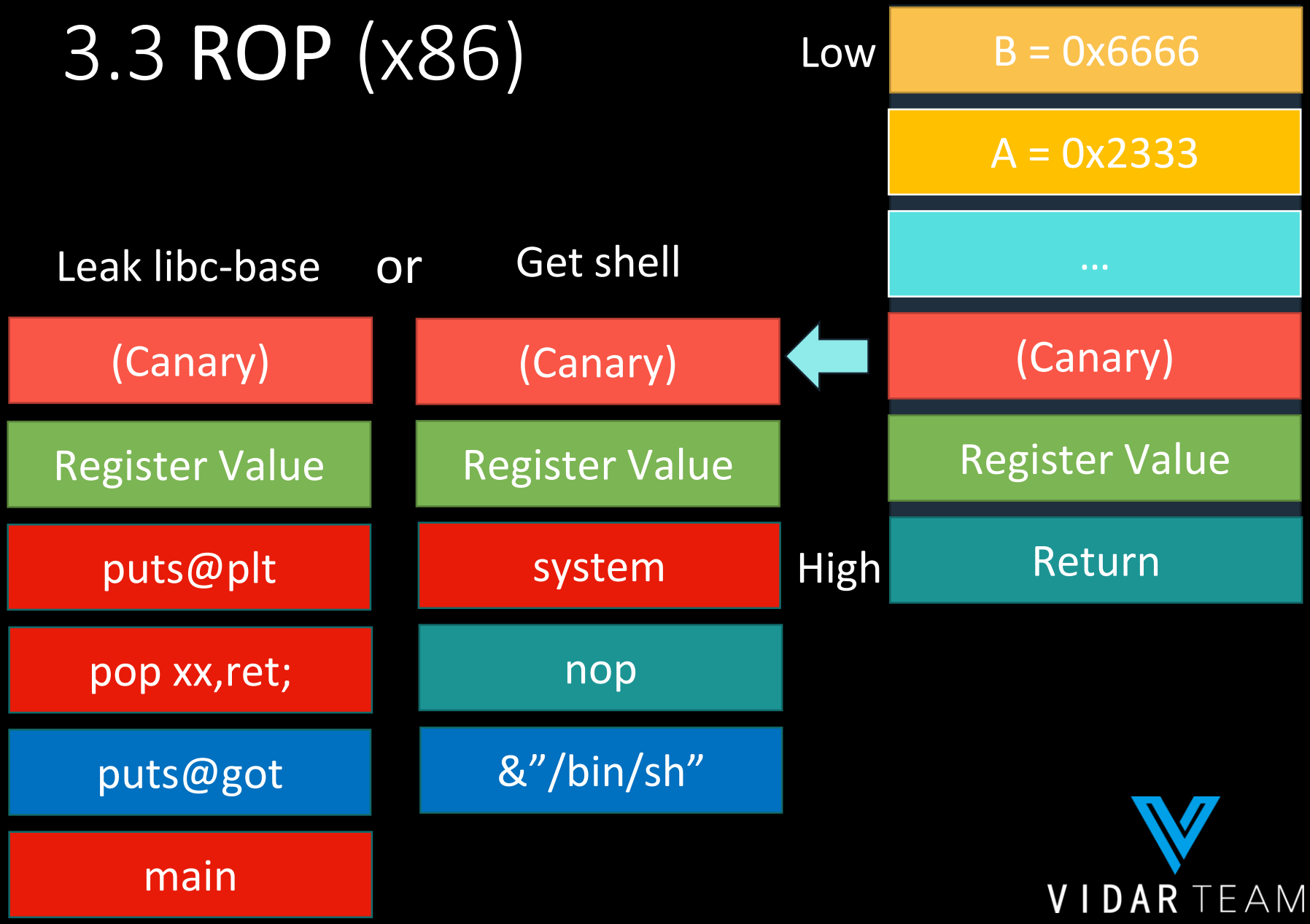
VIDAR TEAM

3.2 ROP (x64)



VIDAR TEAM

3.3 ROP (x86)



VIDAR TEAM

4.1 TOOLS

- IDA Pro -- 反编译工具
- Gdb -- Linux中必要的调试工具
- Pwndbg -- Gdb插件（便于调试）
- Pwntools -- 写exp和poc的利器

VIDAR TEAM

4.1 TOOLS

- Checksec——可以很方便的知道elf程序的安全性和程序的运行平台
- ROPgadget——rop利用工具

……

5.0 PWN!

VIDAR TEAM

5.0 Books!

- 《汇编语言》（王爽）
- 《程序员的自我修养》
- 《加密与解密》

VIDAR TEAM