Reversing 102

從一開始的逆向工程

Inndy / inndy.tw@gmail.com

Recall

- 上星期我們學了：
  - 基礎的幾個常用指令
  - Stack 的運作
  - Calling Convention

04 - fib

- 不開啟任何最佳化
  - gcc -O0
- 開啟不同程度的最佳化
  - 程式會被自動改寫成等效，但是比較快的機器碼
  - gcc -O1 / gcc -O2 / gcc -O3

04 - fib

```
0000000000402de0 <main>:          int main()
  402de0: sub    rsp,0x28         {
  402de4: call   401720 <__main>
  402de9: mov    ecx,0x3            return fib(3) + 3;
  402dee: call   401560 <fib>       // return fib(5) in source code
  402df3: add    eax,0x3          }
  402df6: add    rsp,0x28
  402dfa: ret
```

04 - fib

```
0000000000401560 <fib>:
  401560:   push   rdi
  401561:   push   rsi
  401562:   push   rbx
  401563:   sub    rsp,0x20
  401567:   mov    eax,0x1
  40156c:   cmp    ecx,0x2
  40156f:   jle    40158f <fib+0x2f>
  401571:   lea    edi,[rcx-0x3]
  401574:   lea    ebx,[rcx-0x1]
  401577:   xor    esi,esi
  401579:   and    edi,0x1
  40157c:   mov    ecx,ebx
  40157e:   sub    ebx,0x2
  401581:   call   401560 <fib>
  401586:   add    esi,eax
  401588:   cmp    ebx,edi
  40158a:   jne    40157c <fib+0x1c>
  40158c:   lea    eax,[rsi+0x1]
  40158f:   add    rsp,0x20
  401593:   pop    rbx
  401594:   pop    rsi
  401595:   pop    rdi
  401596:   ret
```

04 - fib

```
0000000000401560 <fib>:
  401567: mov    eax,0x1
  40156c: cmp    ecx,0x2
  40156f: jle    40158f <fib+0x2f>
  401571: lea    edi,[rcx-0x3]
  401574: lea    ebx,[rcx-0x1]
  401577: xor    esi,esi
  401579: and    edi,0x1

  40157c: mov    ecx,ebx
  40157e: sub    ebx,0x2
  401581: call   401560 <fib>
  401586: add    esi,eax
  401588: cmp    ebx,edi
  40158a: jne    40157c <fib+0x1c>
  40158c: lea    eax,[rsi+0x1]
```

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int di = cx - 3;
  int bx = cx - 1;
  int si = 0;
  di &= 1;
  do {
    cx = bx;
    bx -= 2;
    ax = fib(cx);
    si += ax;
  }
  while(bx != di);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int di = cx - 3;
  int bx = cx - 1;
  int si = 0;
  di &= 1;
  do {
    cx = bx;
    bx -= 2;
    ax = fib(cx);
    si += ax;
  }
  while(bx != di);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int di = cx - 3;
  int bx = cx - 1;
  int si = 0;
  di &= 1;
  do {
    cx = bx;
    ax = fib(cx);
    bx -= 2;
    si += ax;
  }
  while(bx != di);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int di = cx - 3;
  int bx = cx - 1;
  int si = 0;
  di &= 1;
  do {
    cx = bx;
    ax = fib(bx);
    bx -= 2;
    si += ax;
  }
  while(bx != di);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int di = cx - 3;
  int bx = cx - 1;
  int si = 0;
  int di = (cx - 3) & 1;
  do {

    ax = fib(bx);
    bx -= 2;
    si += ax;
  }
  while(bx != di);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int bx = cx - 1;
  int si = 0;
  int di = (cx - 3) & 1;
  do {

    ax = fib(bx);
    bx -= 2;
    si += ax;
  }
  while(bx >= 2);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int bx = cx - 1;
  int si = 0;

  do {

    si += fib(bx);
    bx -= 2;

  }
  while(bx >= 2);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;

  int bx = cx - 1;
  int si = 1;

  do {

    si += fib(bx);
    bx -= 2;

  }
  while(bx >= 2);
  return si + 1;
}
```

04 - fib

```
int fib(int cx) {
  int ax = 1;
  if(cx <= 2)
    return ax;
  int bx = cx - 1;
  int si = 1;

  do {

    si += fib(bx);
    bx -= 2;

  }
  while(bx >= 2);
  return si;
}
```

04 - fib

```
int fib(int cx) {
  if(cx <= 2)
    return 1;

  int bx = cx - 1;
  int si = 1;

  do {
    si += fib(bx);
    bx -= 2;
  } while(bx >= 2);

  return si;
}
```

04 - fib

- fib(8) = fib(7) + fib(5) + fib(3) + 1;
- fib(7) = fib(6) + fib(4) + fib(2) + 1;
- fib(6) = fib(5) + fib(3) + 1;
- fib(5) = fib(4) + fib(2) + 1;
- fib(4) = fib(3) + 1;
- fib(3) = fib(2) + 1;

04 - fib

- fib(8) =
- fib(7) + fib(6) =
- fib(7) + fib(5) + fib(4) =
- fib(7) + fib(5) + fib(3) + fib(2) =
- fib(7) + fib(5) + fib(3) + 1

Warm Up

```
func:
cmp rcx, 1
jg calc
mov rax, 1
ret

calc:
push rcx
dec rcx // rcx--
call func
pop rcx
imul rax, rcx // rax *= rcx
ret
```

```
int64_t func(int64_t rcx) {
if(! (rcx > 1))
{
  return 1;
}

int64_t rax;

rax = func(rcx - 1);

return rax * rcx;
}
```

區域變數

- 區域變數會被放在 return address 旁邊的堆疊空間
  - 所以 return 之後就不見了
- sub rsp, XXX
  - 相當於 push 好幾格空間出來
  - 區域變數就存在這裡

Stack Frame

- rbp 指向 stack 中間的某個位置
- 通常就在 return address 旁邊
- 旁邊就是區域變數

Stack Frame

- push rbp
- mov rbp, rsp
- // 這裡開始 function body
- leave // mov rsp, rbp; pop rbp
- ret