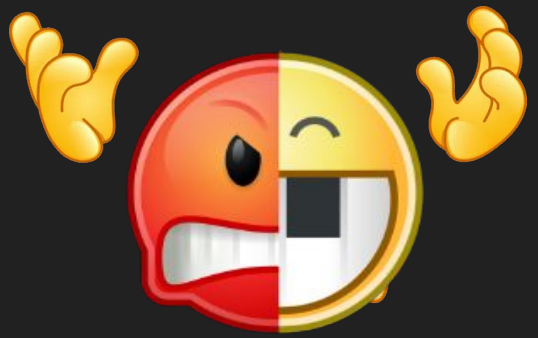


符號執行 :^^

2024/06/30 betan@B33F H4CK1NG

指導老師 : Kazma

Whoami

- betan 貝坦
- 2024 AIS3
- B33F


要上嗎……??
我發表讀書會……!?
betan貝坦
我是考慮欸
被迫上崗
誒…真的假的?
B33F50μP

前情提要

https://kazma.tw/2024/04/30/Angr-CTF-Writeups/

目錄

- 基礎code複習
- 火拳のエース
- 研究xor機制
  - 猜測原因
  - 符號執行介紹
  - 實驗xor
  - 總結

OLD Outline

- review
- ~~scanf~~
- ~~lab003~~
- ~~memory~~
- ~~lab005~~
- ~~lab006~~
- 火拳のエース
- xor

基礎code複習

```python
import angr
import sys

def main(argv):

  path_to_binary = argv[1]
  project = angr.Project(path_to_binary)

  start_address = 0x???
  initial_state = project.factory.blank_state(addr=start_address)

  simulation = project.factory.simgr(initial_state)

  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return state.solver.is_true("Good Job." in str(stdout_output))

  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return state.solver.is_true("Try again." in str(stdout_output))

  simulation.explore(find=is_successful, avoid=should_abort)

  if simulation.found:
    solution_state = simulation.found[0]
    print(solution_state.posix.dumps(sys.stdin.fileno()).decode())
  else:
    raise Exception('Could not find the solution')

if __name__ == '__main__':
  main(sys.argv)
```

基礎code複習

```
path_to_binary = argv[1] # "./chall"
project = angr.Project(path_to_binary)
```

- 載入並建立 angr之檔案
  - $ pipenv run python3 EXP.py chall

基礎code複習

```
start_address = 0x???
initial_state = project.factory.blank_state(addr=start_address)
```

- 設定angr進入地址
- 若無需特別調整則可改為 `entry_state`

基礎code複習

```
simulation = project.factory.simgr(initial_state)
```

- **建立鏡像檔**

基礎code複習

```
def is_successful(state):
  stdout_output = state.posix.dumps(sys.stdout.fileno())
  return state.solver.is_true("Good Job." in str(stdout_output))

def should_abort(state):
  stdout_output = state.posix.dumps(sys.stdout.fileno())
  return state.solver.is_true("Try again." in str(stdout_output))

simulation.explore(find=is_successful, avoid=should_abort)
```

- 定義成功&失敗

基礎code複習

```
if simulation.found:
  solution_state = simulation.found[0]
  print(solution_state.posix.dumps(sys.stdin.fileno()).decode())
else:
  raise Exception('Could not find the solution')
```

- 最後結果判斷

```python
import angr
import sys

def main(argv):

  path_to_binary = argv[1]
  project = angr.Project(path_to_binary)

  start_address = 0x???
  initial_state = project.factory.blank_state(addr=start_address)

  simulation = project.factory.simgr(initial_state)

  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return state.solver.is_true("Good Job." in str(stdout_output))

  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return state.solver.is_true("Try again." in str(stdout_output))

  simulation.explore(find=is_successful, avoid=should_abort)

  if simulation.found:
    solution_state = simulation.found[0]
    print(solution_state.posix.dumps(sys.stdin.fileno()).decode())
  else:
    raise Exception('Could not find the solution')

if __name__ == '__main__':
  main(sys.argv)
```

Lab

https://angr.oregonctf.org/

scanf

```
# Angr doesn't currently support reading multiple things with scanf (Ex: scanf("%u %u).) You will have to tell the simulation engine to begin the program after scanf is called and manually inject the symbols into registers.
```

Lab003

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 user_input; // rax
  int v5; // [esp+4h] [ebp-14h]
  int v6; // [esp+8h] [ebp-10h]
  int v7; // [esp+Ch] [ebp-Ch]
  int v8; // [esp+Ch] [ebp-Ch]

  printf("Enter the password: ");
  user_input = get_user_input();
  v7 = HIDWORD(user_input);
  v5 = complex_function_1(user_input);
  v6 = complex_function_2();
  v8 = complex_function_3(v7);
  if ( v5 || v6 || v8 )
    puts("Try again.");
  else
    puts("Good Job.");
  return 0;
}
```

```
int get_user_input()
{
  int v1; // [esp+0h] [ebp-18h] BYREF
  int v2; // [esp+4h] [ebp-14h] BYREF
  int v3[4]; // [esp+8h] [ebp-10h] BYREF

  v3[1] = __readgsdword(0x14u);
  __isoc99_scanf("%x %x %x", &v1, &v2, v3);
  return v1;
}
```

```python
import angr
import claripy
import sys

def main(argv):
  path_to_binary = argv[1]
  project = angr.Project(path_to_binary,auto_load_libs=False)

  start_address = 0x080488B1
  initial_state = project.factory.blank_state(addr=start_address)

  password_size_in_bits = 32
  password0 = claripy.BVS('password0', password_size_in_bits)
  password1 = claripy.BVS('password1', password_size_in_bits)
  password2 = claripy.BVS('password2', password_size_in_bits)

  initial_state.regs.eax = password0
  initial_state.regs.ebx = password1
  initial_state.regs.edx = password2

  simulation = project.factory.simgr(initial_state)

  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())

    return b"Good Job." in stdout_output

  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b"Try again." in stdout_output

  simulation.explore(find=is_successful, avoid=should_abort)

  if simulation.found:
    solution_state = simulation.found[0]


    solution0 = solution_state.solver.eval(password0)
    solution1 = solution_state.solver.eval(password1)
    solution2 = solution_state.solver.eval(password2)

    solution = ' '.join(map('{:x}'.format, [ solution0, solution1, solution2 ]))
    print (solution)
  else:
    raise Exception('Could not find the solution')

if __name__ == '__main__':
  main(sys.argv)
```

```
.text:08049575                  call    get_user_input
.text:0804957A                  mov     [ebp+var_14], eax
```

```
┌──(kali㉿kali)-[~/Desktop/angr]
└─$ file 02_angr_find_condition
02_angr_find_condition: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynam
e3e7a3634d92fcfbdb4f65246, for GNU/Linux 3.2.0, not stripped
```

```
call    ___isoc99_scanf
add     esp, 10h
mov     eax, [ebp+var_18]
mov     edx, [ebp+var_14]
mov     ebx, edx
mov     edx, [ebp+var_10]
nop
mov     ecx, [ebp+var_C]
xor     ecx, large gs:14h
jz      short locret_804954E
```

memory

- memory
- dynamic memory

Lab005

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int i; // [esp+Ch] [ebp-Ch]
4
5    memset(&user_input, 0, 33);
6    printf("Enter the password: ");
7    __isoc99_scanf("%8s %8s %8s %8s", &user_input, &unk_97EE8E8, &unk_97EE8F0, &unk_97EE8F8);
8    for ( i = 0; i <= 31; ++i )
9      *(_BYTE *)(i + 0x97EE8E0) = complex_function(*(char *)(i + 159312096), i);
10   if ( !strncmp(&user_input, "SRKZHLRMFGQDUCASEFSCWULLFOAWYEDE", 32) )
11     puts("Good Job.");
12   else
13     puts("Try again.");
14   return 0;
15 }
```

```
.bss:097EE8E0 user_input
.bss:097EE8E0
.bss:097EE8E1
.bss:097EE8E2
.bss:097EE8E3
.bss:097EE8E4
.bss:097EE8E5
.bss:097EE8E6
.bss:097EE8E7
.bss:097EE8E8 unk_97EE8E8
.bss:097EE8E9
.bss:097EE8EA
.bss:097EE8EB
.bss:097EE8EC
.bss:097EE8ED
.bss:097EE8EE
.bss:097EE8EF
.bss:097EE8F0 unk_97EE8F0
```

```python
import angr
import claripy
import sys

def main(argv):
  path_to_binary = argv[1]
  project = angr.Project(path_to_binary)
  start_address = 0x08049318
  initial_state = project.factory.blank_state(
    addr=start_address,
    add_options = { angr.options.SYMBOL_FILL_UNCONSTRAINED_MEMORY,
                    angr.options.SYMBOL_FILL_UNCONSTRAINED_REGISTERS}
  )
  password0 = claripy.BVS('password0', 64)
  password1 = claripy.BVS('password1', 64)
  password2 = claripy.BVS('password2', 64)
  password3 = claripy.BVS('password3', 64)

  password0_address = 0x97EE8E0
  initial_state.memory.store(password0_address, password0)
  password1_address = 0x97EE8E8
  initial_state.memory.store(password1_address, password1)
  password2_address = 0x97EE8F0
  initial_state.memory.store(password2_address, password2)
  password3_address = 0x97EE8F8
  initial_state.memory.store(password3_address, password3)

  simulation = project.factory.simgr(initial_state)

  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b'Good Job.' in stdout_output
  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b'Try again.' in stdout_output
  simulation.explore(find=is_successful, avoid=should_abort)
  if simulation.found:
    solution_state = simulation.found[0]
    solution0 = solution_state.solver.eval(password0,cast_to=bytes).decode()
    solution1 = solution_state.solver.eval(password1,cast_to=bytes).decode()
    solution2 = solution_state.solver.eval(password2,cast_to=bytes).decode()
    solution3 = solution_state.solver.eval(password3,cast_to=bytes).decode()
    print (solution0,solution1,solution2,solution3)
  else:
    raise Exception('Could not find the solution')
if __name__ == '__main__':
  main(sys.argv)
```

Lab006

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int i; // [esp+Ch] [ebp-Ch]
4
5    buffer0 = malloc(9);
6    buffer1 = malloc(9);
7    memset(buffer0, 0, 9);
8    memset(buffer1, 0, 9);
9    printf("Enter the password: ");
10   __isoc99_scanf("%8s %8s", buffer0, buffer1);
11   for ( i = 0; i <= 7; ++i )
12   {
13     *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
14     *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
15   }
16   if ( !strncmp(buffer0, "TXMDOLCF", 8) && !strncmp(buffer1, "VIQBYQFD", 8) )
17     puts("Good Job.");
18   else
19     puts("Try again.");
20   free(buffer0);
21   free(buffer1);
22   return 0;
23 }
```

```
.bss:0B5CB7F0 buffer0
.bss:0B5CB7F0
.bss:0B5CB7F4
.bss:0B5CB7F4 buffer3
.bss:0B5CB7F5
.bss:0B5CB7F6
.bss:0B5CB7F7
.bss:0B5CB7F8
.bss:0B5CB7F8 buffer1
```

```python
import angr
import claripy
import sys

def main(argv):
  path_to_binary = argv[1]
  project = angr.Project(path_to_binary)
  start_address = 0X0804938F
  initial_state = project.factory.blank_state(
    addr=start_address,
    add_options = { angr.options.SYMBOL_FILL_UNCONSTRAINED_MEMORY,
                    angr.options.SYMBOL_FILL_UNCONSTRAINED_REGISTERS}
  )
  password0 = claripy.BVS('password0', 64)
  password1 = claripy.BVS('password1', 64)

  fake_heap_address0 = 0xffffc93c
  pointer_to_malloc_memory_address0 = 0x0B5CB7F0
  initial_state.memory.store(pointer_to_malloc_memory_address0, fake_heap_address0, endness=project.arch.memory_endness)

  fake_heap_address1 = 0xffffc94c
  pointer_to_malloc_memory_address1 = 0x0B5CB7F8
  initial_state.memory.store(pointer_to_malloc_memory_address1, fake_heap_address1, endness=project.arch.memory_endness)


  initial_state.memory.store(fake_heap_address0, password0)
  initial_state.memory.store(fake_heap_address1, password1)

  simulation = project.factory.simgr(initial_state)
  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b'Good Job.' in stdout_output
  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b'Try again.' in stdout_output
  simulation.explore(find=is_successful, avoid=should_abort)
  if simulation.found:
    solution_state = simulation.found[0]
    solution0 = solution_state.solver.eval(password0,cast_to=bytes).decode()
    solution1 = solution_state.solver.eval(password1,cast_to=bytes).decode()
    print (solution0, solution1)
  else:
    raise Exception('Could not find the solution')
if __name__ == '__main__':
  main(sys.argv)
```

```
.bss:0B5CB7F0 buffer0
.bss:0B5CB7F0
.bss:0B5CB7F4
.bss:0B5CB7F4 buffer3
.bss:0B5CB7F5
.bss:0B5CB7F6
.bss:0B5CB7F7
.bss:0B5CB7F8
.bss:0B5CB7F8 buffer1
```

BEFORE:

buffer0 -> malloc()ed address 0 -> string 0

buffer1 -> malloc()ed address 1 -> string 1

AFTER:

buffer0 -> fake address 0 -> symbolic bitvector 0

buffer1 -> fake address 1 -> symbolic bitvector 1

火拳のエース

挑战 个解出

火拳のエース

408

rev easy

What I'm about to say. Old Man... Everyone... And you, Luffy... Even though... I'm so worthless... Even though... I carry the blood of a demon... Thank you... For loving me

Author: Kazma

dist-rage-c...

Flag

提交

火拳のエース

```
v3 = time(0, v5, v6, v7, v8, v9, v10, v11);
srand(v3);
buffer0 = malloc(9);
buffer1 = malloc(9);
buffer2 = malloc(9);
buffer3 = malloc(9);
memset(buffer0, 0, 9);
memset(buffer1, 0, 9);
memset(buffer2, 0, 9);
memset(buffer3, 0, 9);
print_flag();
__isoc99_scanf("%8s %8s %8s %8s", buffer0, buffer1, buffer2, buffer3);
xor_strings(buffer0, &unk_804A163, buffer0);
xor_strings(buffer1, v12, buffer1);
xor_strings(buffer2, v13, buffer2);
xor_strings(buffer3, v14, buffer3);
for ( i = 0; i <= 7; ++i )
{
  *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
  *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
  *(_BYTE *)(buffer2 + i) = complex_function(*(char *)(buffer2 + i), i + 64);
  *(_BYTE *)(buffer3 + i) = complex_function(*(char *)(buffer3 + i), i + 96);
}
if ( !strncmp(buffer0, "DHLIYJEG", 8)
  && !strncmp(buffer1, "MZRERYND", 8)
  && !strncmp(buffer2, "RUYODBAH", 8)
  && !strncmp(buffer3, "BKEMPBRE", 8) )
{
  puts("Yes! I remember now, this is it!");
```

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int i; // [esp+Ch] [ebp-Ch]
4
5    buffer0 = malloc(9);
6    buffer1 = malloc(9);
7    memset(buffer0, 0, 9);
8    memset(buffer1, 0, 9);
9    printf("Enter the password: ");
10   __isoc99_scanf("%8s %8s", buffer0, buffer1);
11   for ( i = 0; i <= 7; ++i )
12   {
13     *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
14     *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
15   }
16   if ( !strncmp(buffer0, "TXMDOLCF", 8) && !strncmp(buffer1, "VIQBYQFD", 8) )
17     puts("Good Job.");
18   else
19     puts("Try again.");
20   free(buffer0);
21   free(buffer1);
22   return 0;
23 }
```

print_flag()

```
1  int print_flag()
2  {
3    int i; // [esp+8h] [ebp-10h]
4    int v2; // [esp+Ch] [ebp-Ch]
5
6    printf(
7      "What I'm about to say. Old Man... Everyone... And you, Luffy... Even though... I'm so worthless... Even though... I "
8      "carry the blood of a demon... Thank you... For loving me\n"
9      "The flag is ");
10   v2 = 2000000;
11   usleep(2000000);
12   for ( i = 0; aAis3G0d[i]; ++i )
13   {
14     usleep(v2);
15     printf("%c ", aAis3G0d[i]);
16     fflush(_bss_start);
17     v2 *= 2;
18   }
19   usleep(v2);
20   return puts("\n...uh, the rest, I've forgotten it. Do you remember the rest of it?");
21 }
```

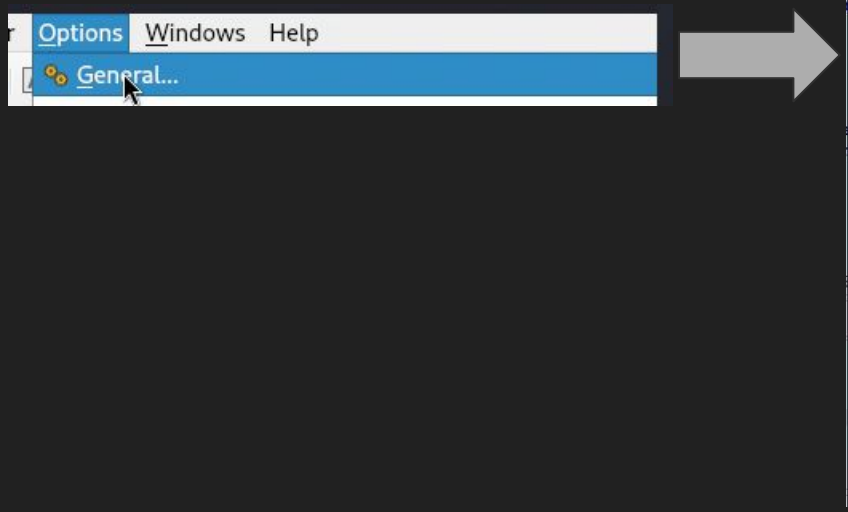
Delete print_flag()

```
push    0
push    eax
call    _memset
add     esp, 10h
call    print_flag
mov     ebx, ds:buffer3
mov     ecx, ds:buffer2
mov     edx, ds:buffer1
mov     eax, ds:buffer0
sub     esp, 0Ch
push    ebx
```

debugger

```
.text:08049629 call    _memset
.text:0804962E add     esp, 10h
.text:08049631 call    print_flag
.text:08049636 mov     ebx, ds:buffer3
.text:0804963C mov     ecx, ds:buffer2
.text:08049642 mov     edx, ds:buffer1
.text:08049648 mov     eax, ds:buffer0
.text:0804964D sub     esp, 0Ch
.text:08049650 push    ebx
```

Delete print_flag()



Delete print_flag()

Before:

```
.text:08049629 call     _memset
.text:0804962E add      esp, 10h
.text:08049631 call     print_flag
.text:08049636 mov      ebx, ds:buffer3
.text:0804963C mov      ecx, ds:buffer2
.text:08049642 mov      edx, ds:buffer1
.text:08049648 mov      eax, ds:buffer0
.text:0804964D sub      esp, 0Ch
.text:08049650 push     ebx
```

After:

```
.text:08049629 E8 B2 FA FF FF     call     _memset
.text:0804962E 83 C4 10           add      esp, 10h
.text:08049631 E8 00 FC FF FF     call     print_flag
.text:08049636 8B 1D E0 B2 0F 09  mov      ebx, ds:buffer3
.text:0804963C 8B 0D DC B2 0F 09  mov      ecx, ds:buffer2
.text:08049642 8B 15 D8 B2 0F 09  mov      edx, ds:buffer1
.text:08049648 A1 D4 B2 0F 09     mov      eax, ds:buffer0
.text:0804964D 83 EC 0C           sub      esp, 0Ch
```

Delete print_flag()

Before:

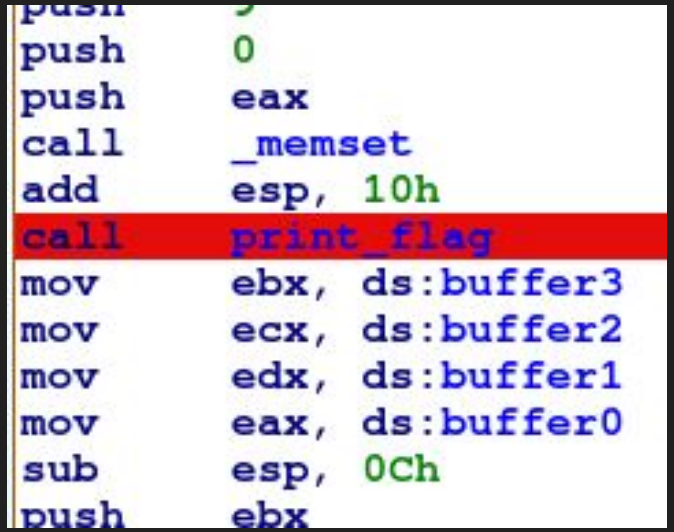
```
1600  DC FA FF FF 83 C4 10 A1 DC B2 0F 09 83 EC 04
1610  09 6A 00 50 E8 C7 FA FF FF 83 C4 10 A1 E0 B2
1620  09 83 EC 04 6A 09 6A 00 50 E8 B2 FA FF FF 83
1630  10 E8 00 FC FF FF 8B 1D E0 B2 0F 09 8B 0D DC
1640  0F 09 8B 15 D8 B2 0F 09 A1 D4 B2 0F 09 83 EC
1650  53 51 52 50 68 53 A1 04 08 E8 A2 FA FF FF 83
1660  20 C7 45 E4 63 A1 04 08 C7 45 E8 6C A1 04 08
1670  45 EC 75 A1 04 08 C7 45 F0 7E A1 04 08 8B 0D
1680  B2 0F 09 8B 55 E4 A1 D4 B2 0F 09 83 EC 04 51
1690  50 E8 01 FE FF FF 83 C4 10 8B 0D D8 B2 0F 09
```

After:

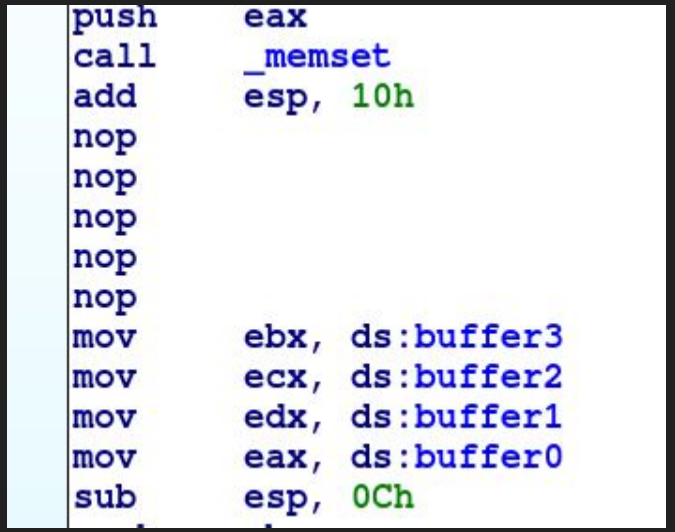
```
1610  09 6A 00 50 E8 C7 FA FF FF 83 C4
1620  09 83 EC 04 6A 09 6A 00 50 E8 B2
1630  10 90 90 90 90 90 8B 1D E0 B2 0F
1640  0F 09 8B 15 D8 B2 0F 09 A1 D4 B2
1650  53 51 52 50 68 53 A1 04 08 E8 A2
1660  20 C7 45 E4 63 A1 04 08 C7 45 E8
1670  45 EC 75 A1 04 08 C7 45 F0 7E A1
1680  B2 0F 09 8B 55 E4 A1 D4 B2 0F 09
1690  50 E8 01 FE FF FF 83 C4 10 8B 0D
```

Delete print_flag()

Before:



After:




Friendship ended with
IDA FREE
Now
GHIDRA
is my
best frien

```python
def main(argv):
  project = angr.Project(path_to_binary)
  start_address = 0X0804965E
  initial_state = project.factory.blank_state(
    addr=start_address,
    add_options = { angr.options.SYMBOL_FILL_UNCONSTRAINED_MEMORY,
                    angr.options.SYMBOL_FILL_UNCONSTRAINED_REGISTERS}
  )
  password0 = claripy.BVS('password0', 64)
  password1 = claripy.BVS('password1', 64)
  password2 = claripy.BVS('password2', 64)
  password3 = claripy.BVS('password3', 64)

  fake_heap_address0 = 0xffffc93c
  pointer_to_malloc_memory_address0 = 0x090FB2D4
  initial_state.memory.store(pointer_to_malloc_memory_address0, fa

  fake_heap_address1 = 0xffffc94c
  pointer_to_malloc_memory_address1 = 0x090FB2D8
  initial_state.memory.store(pointer_to_malloc_memory_address1, fa

  fake_heap_address2 = 0xffffc95c
  pointer_to_malloc_memory_address2 = 0x090FB2DC
  initial_state.memory.store(pointer_to_malloc_memory_address2, fa

  fake_heap_address3 = 0xffffc96c
  pointer_to_malloc_memory_address3 = 0x090FB2E0
  initial_state.memory.store(pointer_to_malloc_memory_address3, fake_he

  initial_state.memory.store(fake_heap_address0, password0)
  initial_state.memory.store(fake_heap_address1, password1)
  initial_state.memory.store(fake_heap_address2, password2)
  initial_state.memory.store(fake_heap_address3, password3)


  simulation = project.factory.simgr(initial_state)
  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b'Yes! I remember now, this is it!' in stdout_output

  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    return b'It feels slightly wrong, but almost correct...' in stdout_output
```

```
.bss:090FB2D4                 public buffer0
.bss:090FB2D4 buffer0         dd ?
.bss:090FB2D4
.bss:090FB2D8                 public buffer1
.bss:090FB2D8 buffer1         dd ?
.bss:090FB2D8
.bss:090FB2DC                 public buffer2
.bss:090FB2DC buffer2         dd ?
.bss:090FB2DC
.bss:090FB2E0                 public buffer3
.bss:090FB2E0 buffer3         dd ?
.bss:090FB2E0
```

```
if ( !strncmp(buffer0, "DHLIYJEG", 8)
  && !strncmp(buffer1, "MZRERYND", 8)
  && !strncmp(buffer2, "RUYODBAH", 8)
  && !strncmp(buffer3, "BKEMPBRE", 8) )
{
  puts("Yes! I remember now, this is it!");
}
else
{
  puts("It feels slightly wrong, but almost correct...");
}
```

Could not find?

```
┌──(kali㉿kali)-[~/Desktop]
└─$ pipenv run python3 EXP.py change
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0×ffffc93c. Default to arch.bits.
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0×ffffc94c. Default to arch.bits.
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0×ffffc95c. Default to arch.bits.
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0×ffffc96c. Default to arch.bits.
Traceback (most recent call last):
  File "/home/kali/Desktop/EXP.py", line 62, in <module>
    main(sys.argv)
  File "/home/kali/Desktop/EXP.py", line 60, in main
    raise Exception('Could not find the solution')
Exception: Could not find the solution
```

```
v3 = time(0, v5, v6, v7, v8, v9, v10, v11);
srand(v3);
buffer0 = malloc(9);
buffer1 = malloc(9);
buffer2 = malloc(9);
buffer3 = malloc(9);
memset(buffer0, 0, 9);
memset(buffer1, 0, 9);
memset(buffer2, 0, 9);
memset(buffer3, 0, 9);
print_flag();
__isoc99_scanf("%8s %8s %8s %8s", buffer0, buffer1, buffer2, buffer3);
xor_strings(buffer0, &unk_804A163, buffer0);
xor_strings(buffer1, v12, buffer1);
xor_strings(buffer2, v13, buffer2);
xor_strings(buffer3, v14, buffer3);
for ( i = 0; i <= 7; ++i )
{
  *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
  *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
  *(_BYTE *)(buffer2 + i) = complex_function(*(char *)(buffer2 + i), i + 64);
  *(_BYTE *)(buffer3 + i) = complex_function(*(char *)(buffer3 + i), i + 96);
}
if ( !strncmp(buffer0, "DHLIYJEG", 8)
  && !strncmp(buffer1, "MZRERYND", 8)
  && !strncmp(buffer2, "RUYODBAH", 8)
  && !strncmp(buffer3, "BKEMPBRE", 8) )
{
  puts("Yes! I remember now, this is it!");
```

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int i; // [esp+Ch] [ebp-Ch]
4
5    buffer0 = malloc(9);
6    buffer1 = malloc(9);
7    memset(buffer0, 0, 9);
8    memset(buffer1, 0, 9);
9    printf("Enter the password: ");
10   __isoc99_scanf("%8s %8s", buffer0, buffer1);
11   for ( i = 0; i <= 7; ++i )
12   {
13     *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
14     *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
15   }
16   if ( !strncmp(buffer0, "TXMDOLCF", 8) && !strncmp(buffer1, "VIQBYQFD", 8) )
17     puts("Good Job.");
18   else
19     puts("Try again.");
20   free(buffer0);
21   free(buffer1);
22   return 0;
23 }
```

Delete xor_strings()

```
13  v3 = time(0, v5, v6, v7, v8, v9, v10, v11);
14  srand(v3);
15  buffer0 = malloc(9);
16  buffer1 = malloc(9);
17  buffer2 = malloc(9);
18  buffer3 = malloc(9);
19  memset(buffer0, 0, 9);
20  memset(buffer1, 0, 9);
21  memset(buffer2, 0, 9);
22  memset(buffer3, 0, 9);
23  __isoc99_scanf("%8s %8s %8s %8s", buffer0, buffer1, buffe
24  for ( i = 0; i <= 7; ++i )
25  {
26    *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
27    *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
28    *(_BYTE *)(buffer2 + i) = complex_function(*(char *)(buffer2 + i), i + 64);
29    *(_BYTE *)(buffer3 + i) = complex_function(*(char *)(buffer3 + i), i + 96);
30  }
31  if ( !strncmp(buffer0, "DHLIYJEG", 8)
32    && !strncmp(buffer1, "MZRERYND", 8)
33    && !strncmp(buffer2, "RUYODBAH", 8)
34    && !strncmp(buffer3, "BKEMPBRE", 8) )
35  {
36    puts("Yes! I remember now, this is it!");
37  }
38  else
39  {
```

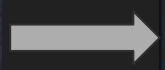
```
v3 = time(0, v5, v6, v7, v8, v9, v10, v11);
srand(v3);
buffer0 = malloc(9);
buffer1 = malloc(9);
buffer2 = malloc(9);
buffer3 = malloc(9);
memset(buffer0, 0, 9);
memset(buffer1, 0, 9);
memset(buffer2, 0, 9);
memset(buffer3, 0, 9);
print_flag();
__isoc99_scanf("%8s %8s %8s %8s", buffer0, buffer1, buffer2, buffer3);
xor_strings(buffer0, &unk_804A163, buffer0);
xor_strings(buffer1, v12, buffer1);
xor_strings(buffer2, v13, buffer2);
xor_strings(buffer3, v14, buffer3);
for ( i = 0; i <= 7; ++i )
{
  *(_BYTE *)(buffer0 + i) = complex_function(*(char *)(buffer0 + i), i);
  *(_BYTE *)(buffer1 + i) = complex_function(*(char *)(buffer1 + i), i + 32);
  *(_BYTE *)(buffer2 + i) = complex_function(*(char *)(buffer2 + i), i + 64);
  *(_BYTE *)(buffer3 + i) = complex_function(*(char *)(buffer3 + i), i + 96);
}
if ( !strncmp(buffer0, "DHLIYJEG", 8)
  && !strncmp(buffer1, "MZRERYND", 8)
  && !strncmp(buffer2, "RUYODBAH", 8)
  && !strncmp(buffer3, "BKEMPBRE", 8) )
{
  puts("Yes! I remember now, this is it!");
```

Success!

```
┌──(kali㉿kali)-[~/Desktop]
└─$ date && pipenv run python3 EXP.py rage002 && date
Tue Jun 11 01:57:15 AM EDT 2024
WARNING  | 2024-06-11 01:57:17,665 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0xffffc93c. Default to arch.bits.
WARNING  | 2024-06-11 01:57:17,666 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0xffffc94c. Default to arch.bits.
WARNING  | 2024-06-11 01:57:17,666 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0xffffc95c. Default to arch.bits.
WARNING  | 2024-06-11 01:57:17,666 | angr.storage.memory_mixins.bvv_conversion_mixin | Unknown size for memory data 0xffffc96c. Default to arch.bits.
QIIKAHBJ TRDMYLWD NMTLWVYB ERHAXFUN
Tue Jun 11 01:58:57 AM EDT 2024
```

Flag

```
gef➤  b *(main+0x12C)
Breakpoint 1 at 0x8049696
gef➤  b *(main+0x148)
Breakpoint 2 at 0x80496b2
gef➤  b *(main+0x164)
Breakpoint 3 at 0x80496ce
gef➤  b *(main+0x180)
Breakpoint 4 at 0x80496ea
```



```
$esp   : 0xffffcf50  →  0x090fc1a0  →  "_D4MN_4N"
$esp   : 0xffffcf50  →  0x090fc1b0  →  "9R_15_5U"
$esp   : 0xffffcf50  →  0x090fc1c0  →  "P3R_P0W3"
$esp   : 0xffffcf50  →  0x090fc1d0  →  "RFU1!!!}"
$ebp   : 0xffffcf88  →  0x00000000
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ strings rage | grep AIS3
AIS3{G0D
```

Flag:

AIS3{G0D_D4MN_4N9R_15_5UP3R_P0W3RFU1!!!}

研究xor機制

xor_strings()

_D4MN_4N □ QIIKAHBKJ

```
1  int __cdecl xor_strings(int a1, int a2, int a3)
2  {
3    _BYTE *v3; // ebx
4    int result; // eax
5    int v5; // [esp+8h] [ebp-20h] BYREF
6    int v6; // [esp+Ch] [ebp-1Ch] BYREF
7    char v7[3]; // [esp+13h] [ebp-15h] BYREF
8    char v8[3]; // [esp+16h] [ebp-12h] BYREF
9    char v9[3]; // [esp+19h] [ebp-Fh] BYREF
10   int i; // [esp+1Ch] [ebp-Ch]
11
12   for ( i = 0; i <= 7; ++i )
13   {
14     char_to_hex(*(char *)(i + a1), v9);
15     char_to_hex(*(char *)(i + a2), v8);
16     __isoc99_sscanf(v9, "%x", &v6);
17     __isoc99_sscanf(v8, "%x", &v5);
18     sprintf(v7, "%02x", v6 ^ v5);
19     v3 = (_BYTE *)(i + a3);
20     *v3 = hex_to_char(v7);
21   }
22   result = a3 + 8;
23   *(_BYTE *)(a3 + 8) = 0;
24   return result;
25 }
```

xor_strings()

四個輸入分別與下列 key做xor

```
key1 = (b'\x0e\x0d\x7d\x06\x0f\x17\x76\x04 ')
key2 = (b'\x6d\x00\x1b\x7c\x6c\x13\x62\x11 ')
key3 = (b'\x1e\x7e\x06\x13\x07\x66\x0e\x71 ')
key4 = (b'\x17\x14\x1d\x70\x79\x67\x74\x33 ')
```

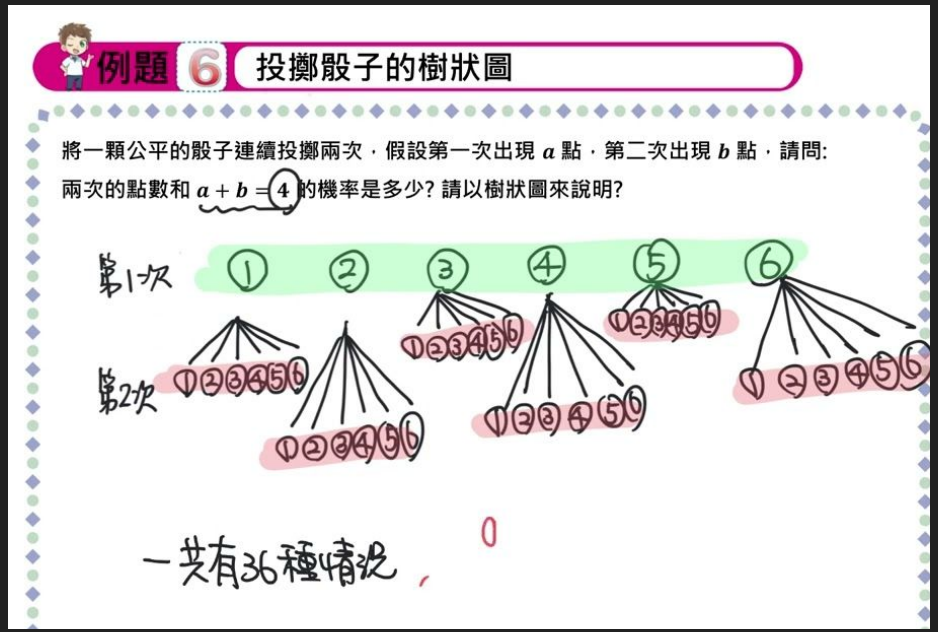
猜測原因

Why?

- 路徑爆炸？
- 其他原因？

甚麼是路徑爆炸

- 由於分支太多，因而導致執行速度緩慢
- 聽不懂？



路徑爆炸

- Will be "No result"

  instead of 'Could not find the solution'

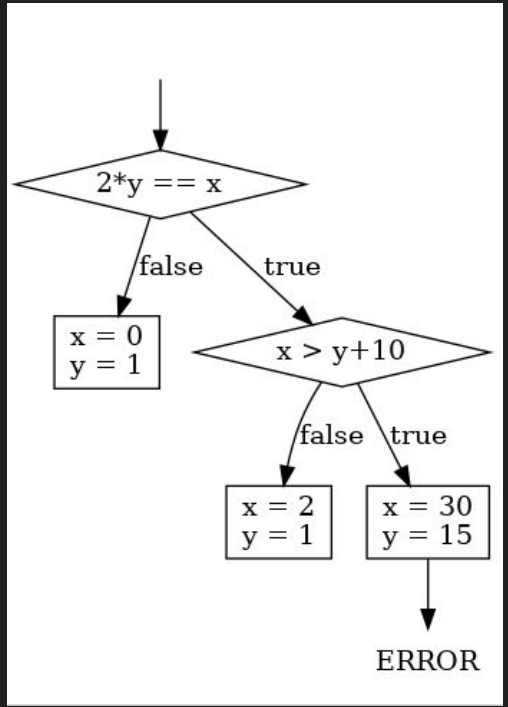
```
┌──(kali㉿kali)-[~/Desktop]
└─$ pipenv run python3 EXP.py change
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_
WARNING  | 2024-06-11 01:32:33,221 | angr.storage.memory_
Traceback (most recent call last):
  File "/home/kali/Desktop/EXP.py", line 62, in <module>
    main(sys.argv)
  File "/home/kali/Desktop/EXP.py", line 60, in main
    raise Exception('Could not find the solution')
Exception: Could not find the solution
```

符號執行介紹

符號執行

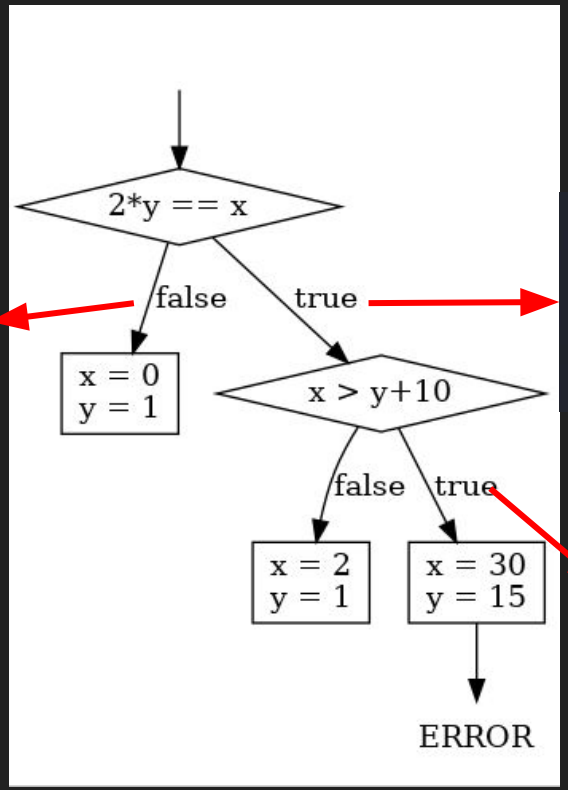
- 設變數配合約束求解
- 在每個分支最後都 產生一個結果
- 結果每次執行可能都不一樣

[angr document]



符號執行

```
>>> state.solver.add(y*2≠x)
[<Bool y_49_64 * 0x2 ≠ x_48_64>]
>>> state.solver.eval(x)
0xffffffffffffffff
>>> state.solver.eval(y)
0x0
```




```
>>> state.solver.add(y*2==x)
[<Bool y_49_64 * 0x2 == x_48_64>]
>>> state.solver.eval(x)
0x0
>>> state.solver.eval(y)
0x0
```

```
>>> state.solver.add(x>y+10)
[<Bool x_48_64 > y_49_64 + 0xa>]
>>> state.solver.eval(x)
0x2e
>>> state.solver.eval(y)
0x17
```

實驗xor

Symbolic Expressions

```
Python 3.11.9 (main, Apr 10 2024, 13:16:36) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import angr, monkeyhex
>>> import claripy
>>>
>>> path_to_binary = "./rage"
>>> proj = angr.Project(path_to_binary)
>>> state = proj.factory.entry_state()
>>> x = claripy.BVS('x', 64)
>>>
>>> x
<BV64 x_17_64>
```

Constraint Solving

perform an operation on bitvectors of differing lengths.

```
>>> state.solver.add(x[0] ^ 0xe == '_')
WARNING  | 2024-06-27 08:59:50,034 | claripy.ast.bv | BVV value is being coerced fr
om a unicode string, encoding as utf-8
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/kali/.local/share/virtualenvs/Desktop-YbM9NXtB/lib/python3.11/site-pa
ckages/claripy/operations.py", line 54, in _op
    raise ClaripyOperationError(msg)
claripy.errors.ClaripyOperationError: args' length must all be equal
>>> x[0]
<BV1_x_17_64[0:0]>
```

通靈時間

- 如果一個的位元是 1，那八個拼在一起呢？

```
>>> one = x[0]+x[1]+x[2]+x[3]+x[4]+x[5]+x[6]+x[7]
>>> one
<BV1 x_17_64[0:0] + x_17_64[1:1] + x_17_64[2:2] + x_17_64[3:3] + x_17_64[4:4] + x_1
7_64[5:5] + x_17_64[6:6] + x_17_64[7:7]>
>>> state.solver.add(one ^ 0xe == '_')
WARNING  | 2024-06-27 09:04:11,416 | claripy.ast.bv | BVV value is being coerced fr
om a unicode string, encoding as utf-8
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/kali/.local/share/virtualenvs/Desktop-YbM9NXtB/lib/python3.11/site-pa
```

通靈時間

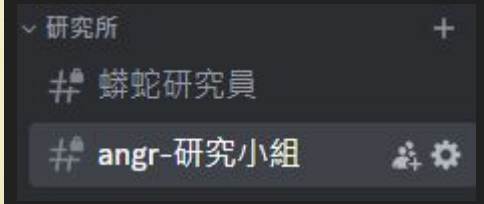
- 那再如果八個位元為 8的符號拼在一起呢？

```
>>> a= claripy.BVS('x', 8)
>>> b= claripy.BVS('x', 8)
>>> c= claripy.BVS('x', 8)
>>> d= claripy.BVS('x', 8)
>>> e= claripy.BVS('x', 8)
>>> f= claripy.BVS('x', 8)
>>> g= claripy.BVS('x', 8)
>>> h= claripy.BVS('x', 8)
>>> a
<BV8 x_19_8>
```

```
>>> ans=a+b+c+d+e+f+g+h
>>> ans
<BV8 x_19_8 + x_20_8 + x_21_8 + x_22_8 + x_23_8 + x_24_8 + x_25_8 + x_26_8>
>>>
>>> state.solver.add(ans[0] ^ 0×e == '_')
WARNING  | 2024-06-27 09:30:38,753 | claripy.ast.bv | BVV value is being coerced fr
om a unicode string, encoding as utf-8
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/kali/.local/share/virtualenvs/Desktop-YbM9NXtB/lib/python3.11/site-pa
ckages/claripy/operations.py", line 54, in _op
    raise ClaripyOperationError(msg)
claripy.errors.ClaripyOperationError: args' length must all be equal
>>>
>>> ans[0]
<BV1 x_19_8 + x_20_8 + x_21_8 + x_22_8 + x_23_8 + x_24_8 + x_25_8 + x_26_8[0:0]>
```

難道...真的沒有辦法變更符號的長度嗎

~~對。~~ 嗎？



Constraint Solving

key1 = (b'\x0e\x0d\x7d\x06\x0f\x17\x76\x04')

ans = '_D4MN_4N'

input = 'QIIKAHBJ'

```
>>> z = claripy.BVS('z', 8)
>>> state.solver.add(z^ 0x0e == '_')
WARNING  | 2024-06-22 07:00:16,886 |
[<Bool (z_19_8 ^ 14) == 95>]
>>> state.solver.eval(x)
0x0
>>> state.solver.eval(z)
0x51
>>> chr(0x51)
'Q'
```

zero_extend

"`zero_extend`" will pad the bitvector on the left with the given number of zero bits.

[angr document]

```
>>> x
<BV64 x_17_64>
>>> z
<BV8 z_19_8>
>>> x.zero_extend(-8)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/kali/.local/share/virtualenvs/kali-_cbpxKIc/lib/python3.11/site-packages/claripy/ast/bv.py", line 127, in zero_extend
    return ZeroExt(n, self)
           ^^^^^^^^^^^^^^^^
  File "/home/kali/.local/share/virtualenvs/kali-_cbpxKIc/lib/python3.11/site-packages/claripy/operations.py", line 54, in _op
    raise ClaripyOperationError(msg)
claripy.errors.ClaripyOperationError: Extension length must be nonnegative
>>> x.zero_extend(8)
<BV72 0#8 .. x_17_64>
```

Constraint Solving

```
>>> key=b'\x0e\x0d\x7d\x06\x0f\x17\x76\x04'
>>> key
b'\x0e\r}\x06\x0f\x17v\x04'
>>> state.solver.add(x ^ key == '_D4MN_4N')
WARNING  | 2024-06-22 07:14:21,988 | claripy.ast.bv | BVV value
[<Bool (x_17_64 ^ 0×e0d7d060f177604) == 0×5f44344d4e5f344e>]
>>> state.solver.eval(x)
0×5149494b4148424a
```

```
>>> ans
'QIIKAHBJ'
>>> for i in range(8):
...     ord(ans[i])
...
0×51
0×49
0×49
0×4b
0×41
0×48
0×42
0×4a
```

總結

總結

- 正常的xor是可以執行的
- 符號無法被分割
- ~~八個臭皮匠還是八個臭皮匠~~

Thank You