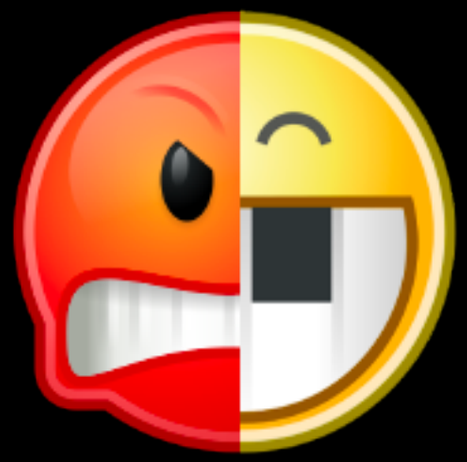
Angr 簡單用用看

講者：OAlienO

angr

angr is a python framework for analyzing binaries. It combines both static and dynamic symbolic ("concolic") analysis, making it applicable to a variety of tasks.

https://docs.angr.io/INSTALL.html



http://angr.io/

How to use anger

直接來寫題目吧

ais3_crackme

能 verify 成功的 argv[1] 就是 flag

```
if ( argc == 2 )
{
  if ( (unsigned int)verify(argv[1], argv, envp) )
    puts("Correct! that is the secret key!");
  else
    puts("I'm sorry, that's the wrong secret key!");
  result = 0;
}
else
{
  puts("You need to enter the secret key!");
  result = -1;
}
```

https://github.com/angr/angr-doc/tree/master/examples/ais3_crackme

ais3_crackme

```
import angr
import claripy

project = angr.Project("./ais3_crackme")
```

目標程式

ais3_crackme

用 IDA 找出長度

宣告一個 symbolic 符號

```
length = 23
argv1 = claripy.BVS("argv1", length * 8)
state = project.factory.entry_state(args = ["./ais3_crackme", argv1])
```

```
oalieno@oalieno ~/Alien/angr-doc/examples/ais3_crackme master ./ais3_crackme hello
I'm sorry, that's the wrong secret key!
```

ais3_crackme

project.factory.entry_state 會回傳一個 SimState 的類別
裡面包括了 register memory 的狀態

```
>>> state.regs.rax
<BV64 0x1c>

>>> state.mem[0x400410].int
<int (32 bits) <BV32 0x8949ed31> at 0x400410>
```

ais3_crackme

simulation manager 負責做 symbolic execution

↓

```
simgr = project.factory.simgr(state)
simgr.explore(find = 0x400602, avoid = 0x40060E)
```

find : 找到這個

↓

```
400602          mov     edi, offset aCorrectThatIsT ; "Correct! that is the secret key!"
40060E          mov     edi, offset aIMSorryThatSTh ; "I'm sorry, that's the wrong secret key!"
```

↑

avoid : 看到這個就不要在繼續了

ais3_crackme

在這個 state 的情況下 argv1 是多少

↓

print simgr.found[0].solver.eval(argv1, cast_to = str)

simgr explore 結束後 simgr 會把有用的 state 分類
變成好多堆的 stash
stash 的種類有分 found, avoid, active, deadended
stash 裡面存了很多的 state