FCSC Chaussette – A Triton showcase

Victor « deadc0de » Houal

Quarkslab

SOMMAIRE

TRITON C'EST QUOI ?

- Framework d'exécution concolique
  - Un moteur d'exécution symbolique
  - Un moteur de teinte
  - Des interfaces vers Z3, Bitwuzla et LLVM
  - Une API Python / C++

TRITON C'EST QUOI ?

```python
from triton import *

# Créer le contexte Triton avec une architecture définie
ctx = TritonContext(ARCH.X86_64)

# Définir des valeurs concrètes (facultatif)
ctx.setConcreteRegisterValue(ctx.registers.rip, 0x1000)

# Symboliser les données (facultatif)
ctx.symbolizeRegister(ctx.registers.rax, 'my_rax')

# Exécuter les instructions
ctx.processing(Instruction(b"\x48\x35\x34\x12\x00\x00")) # xor rax, 0x1234
ctx.processing(Instruction(b"\x48\x89\xc1")) # mov rcx, rax

# Obtenir l'expression symbolique
rcx_expr = ctx.getSymbolicRegister(ctx.registers.rcx)
print(rcx_expr)
# (define-fun ref!8 () (_ BitVec 64) ref!1) ; Mov Operation - 0x40006: mov rcx, rax

# Résoudre la contrainte
ctx.getModel(rcx_expr.getAst() == 0xdead)
#{0: my_rax:64 = 0xcc99}

# 0xcc99 XOR 0x1234 est en effet égal à 0xdead
hex(0xcc99 ^ 0x1234)
# '0xdead'
```

CHAUSSETTE, MAIS C'EST TOUT SIMPLE

```
3
4   int main( int argc, char *argv[], char *envp[]) {
5
6     unsigned long  v5,v6,v7;
7     unsigned long input1 = 0;
8     unsigned long input2 = 0;
9
10    scanf("%lu",&input1);
11    scanf("%lu",&input2);
12    /*
13      - skip condition sysconf
14      - skip mmap void * generic_pt_mmap = mmap(...)
15
16    */
17   if (MBA_Function(input1,input2) && mprotect(addr, 0x400, PROT_READ | PROT_EXEC) ++ -1 ) {
18      v5 = input1;
19      v6 = &shellCode;
20      v7 = input2;
21      do
22      {
23        *v6 ^= v7;
24        v6 += 8;
25        v7 = 0x5851F42D4C957F2D * v7 + 0x14057B7EF767814F;
26
27      } while ((v6 - &shellCode) <= 516);
28      (shellcode())(generic_ptr_mmap, v5, lu2)
29      /*
30        getprotobyname();
31        socket();
32        gethostbyname();
33        inet_ntoa();
34        inet_addr();
35        connect();
36        write();
37        read();
38        scanf("%lu",&input1);
39        scanf("%lu",&input2);
40      */
41   }
42
43   }
```

CHAUSSETTE, MAIS C'EST TOUT SIMPLE

```
unsigned long mba(unsigned long input1, unsigned long input1)
{
  return (0xC88D84433F7B14E7
        * (1
        - __ROL8__(
            __ROL8__(
              __ROL8__(
                __ROR8__(
                  0x9F40CB3B786D6843
                * (__ROR8__(
                    -(__ROR8__(
                      0x7F7965E1BACA6C90
                    - ((0x94224BCAD3296113 * (input1 - 0x33A11A22D2EA22D2) - 0x157C4AC14733D2A2) ^ 0x5819EA9FCBC8779),
                      13)
                    - 0x4368955C512CA39B
                    + 1),
                    50)
                 + 0x693348A42A967F84),
                  20)
              + 0x4596C7E1C75794F7,
                54)
            + 1,
              33),
            56))) ^ 0xB12E2565D1EFE9F8;
}
```

CHAUSSETTE, MAIS C'EST TOUT SIMPLE

```python
from triton import *

def emulate(ctx, pc = 0x1000):

  while pc:
    opcode = ctx.getConcreteMemoryAreaValue(pc, 16) # on ne connait pas la taille de l'instruction qui vient, on lit donc le maxsize :)
    instruction = Instruction()
    instruction.setOpcode(opcode)
    instruction.setAddress(pc)
    ctx.processing(instruction)
    print(instruction)

    pc = ctx.getConcreteRegisterValue(ctx.registers.rip)

ctx = TritonContext()
ctx.setArchitecture(ARCH.X86_64)

obfu = codeDumped

pc = 0x1000
ctx.setConcreteMemoryAreaValue(pc,obfu)

ctx.symbolizeRegister(ctx.registers.rdi) # On symbolise rdi & rsi
ctx.symbolizeRegister(ctx.registers.rsi)

emulate(ctx,pc)
rax = ctx.getSymbolicRegister(ctx.registers.rax) # result is in rax
# 0x005555555560B0                test    rax, rax
print(ctx.getModel(rax.getAst() == 0x0))

# output : Solve RAX == 0x0
```

LES PROBLÈMES

- La taille de l'expression à résoudre grandit à chaque opération

```
┌─[deadc0de@tacosServer2:~/F/chaussetteF]─[23:59:18]
└─>$ ls -S -l | awk '/^-/{print "Size: " $5 " bytes | File: " $9}' | tac

Size: 929 bytes | File: 0.bin
Size: 1889 bytes | File: 1.bin
Size: 4173 bytes | File: 2.bin
Size: 8463 bytes | File: 3.bin
Size: 16514 bytes | File: 4.bin
Size: 27643 bytes | File: 7.bin
Size: 33081 bytes | File: 5.bin
Size: 56107 bytes | File: 8.bin
Size: 67866 bytes | File: 6.bin
Size: 111887 bytes | File: 9.bin
Size: 304961 bytes | File: _10.bin
```

DIFFÉRENTE MANIÈRE D'ATTAQUER

- Pattern Matching
- Neural Networks
- Bit-blasting (Arybo)
- Stochastic program synthesis (Syntia)
- Synthesis-based expression simplification (Qsynth)

LES PROBLÈMES

- QSynthesis est une API Python3 qui permet de simplifier des expressions arithmétiques booléennes mixée.

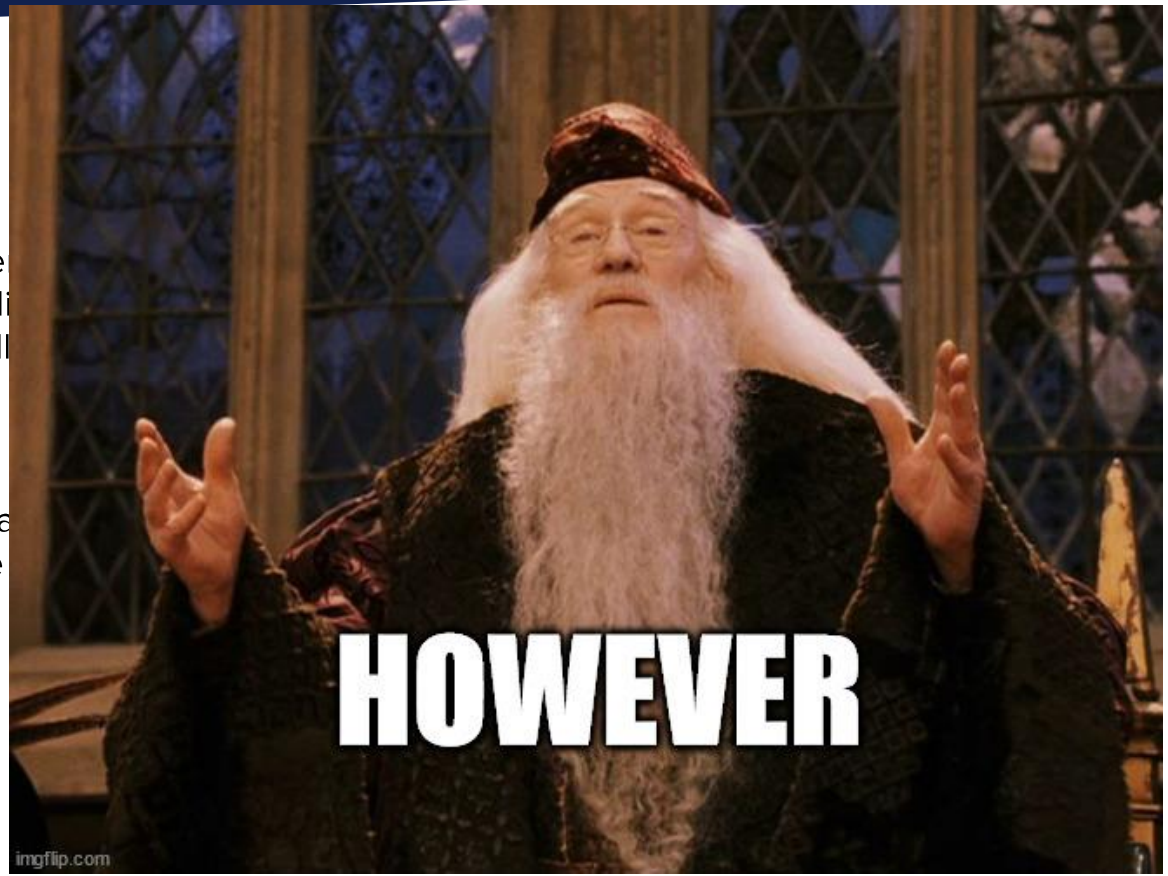
```
int f(unsigned a, unsigned b) {
    unsigned n = (a & ~(-((((b & a) * (b | a) + (b & ~a) * (~b & a)) & b) *                  \
                  (((b&a)*(b|a) + (b& ~a)*(~b&a))|b) + (((b&a)*(b| a) + (b & ~a) *          \
                  (~b & a)) & ~b) * (~((b & a) * (b | a) + (b & ~a) * (~b & a)) & b)) - 1)) -  \
                  (~a & (-((((b & a) * (b | a) + (b & ~a) * (~b & a)) & b) * (((b & a) *    \
                  (b | a) + (b & ~a) * (~b & a)) | b) + (((b & a) * (b | a) + (b & ~a) *     \
                  (~b & a)) & ~b) * (~((b & a) * (b | a) + (b & ~a) * (~b & a)) & b)) - 1));  \
    return n;
}
```


```
int f(unsigned a, unsigned b) {
    unsigned n = a - (~((b * a) * b));
    return n;
}
```

LES PROBLÈMES

POURQUOI TRITON NE FONCTIONNAIT PAS

- La conve
  intermédi
  potentiell
- PBKAC
- L'auteur a
  ce genre



- Triton pouvait le faire :
  - Bitwuzla
  - Triton optimisation

```
from triton import *


ctx = TritonContext(ARCH.X86_64)

ctx.setMode(MODE.CONSTANT_FOLDING, True)
ctx.setMode(MODE.AST_OPTIMIZATIONS, True)
ctx.setMode(MODE.ONLY_ON_SYMBOLIZED, True)



ctx.setSolver(SOLVER.BITWUZLA) # work with this
```

Thank you !

- Triton : https://github.com/JonathanSalwan/Triton
- Qsynthesis : https://github.com/quarkslab/qsynthesis

Bibliographie :

- https://blog.quarkslab.com/resources/2017-06-09-nouthese-soutenance/Obfuscation_with_Mixed_Boolean_Arithmetic_Expressions_Eyrolles.pdf

- https://github.com/JonathanSalwan/Triton/blob/master/publications/BAR2020-qsynth-robin-david.pdf

- https://bitwuzla.github.io/data/fmcad2020/NiemetzPreiner-FMCAD20.pdf
- https://github.com/DenuvoSoftwareSolutions/GAMBA/blob/main/paper/paper.pdf

Quarkslab