FA2022 Week 08

Reverse Engineering II

Richard

Announcements

- ACM clean up round 3
  - Now an official ACM social event!!
  - After this Sunday's meeting!!!
  - With pizza!!!!!!
- Sunday seminar: guest speaker Mingjia
  - Sensitive healthcare data & third party trackers

ctf.sigpwny.com

sigpwny{4ngr_g0_brrrrrrrrr}

Table of Contents

Constraint solving

Constraint solving

- Solve complex systems of equations
- z3
  - python library for solving constraints
  - `pip install z3-solver`

```
if (input_arr[15] == 91.0) {
  if (input_arr[18] == 91.0) {
    if (input_arr[0] + input_arr[0] + 11.0 == input_arr[0] + 130.0) {
      if (input_arr[23] + input_arr[23] + 6.0 == input_arr[23] + 127.0) {
        if (input_arr[1] * 7.0 == input_arr[1] + 396.0) {
          if (input_arr[22] == 104.0) {
            if ((input_arr[2] + 2.0) * 3.0 - 2.0 == (input_arr[2] - 17.0) * 4.0) {
              if (input_arr[21] == (input_arr[21] + input_arr[21]) - 44.0) {
                if (input_arr[3] == 67.0) {
                  if ((input_arr[20] * 3.0 - 2.0) * 3.0 - (input_arr[20] * 5.0 + 2.0) * 4.0
                      == input_arr[20] * -8.0 - 146.0) {
                    if ((input_arr[4] * 5.0 - 2.0) * 5.0 -
                        (input_arr[4] + input_arr[4] + 7.0) * 6.0 ==
                        input_arr[4] * 33.0 - 1132.0) {
```

z3 example

```python
from z3 import *

# define variables
x = Int('x')
y = Int('y')

# add constraints
s = Solver()
s.add(x + y == 12)
s.add(x < y)

print(s.check()) # prints "sat" if has solution

# print solution
m = s.model()
print(m[x])
print(m[y])
```

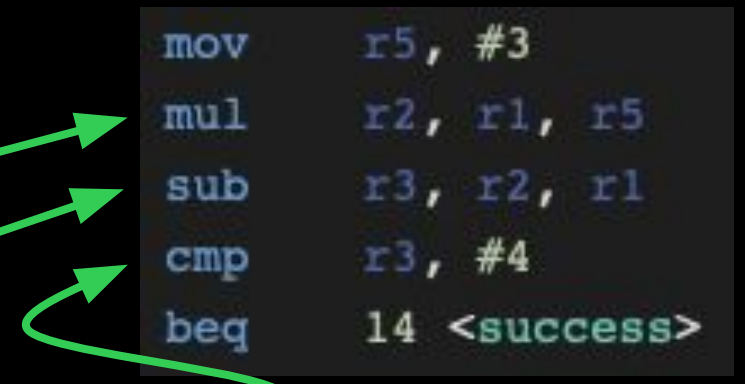
$$\begin{cases} x + y = 12 \\ x < y \end{cases}$$

(Note: this finds any of the possible solutions)

Symbolic Execution

- Solve for inputs
  - Generate constraints from program **automatically**

```
x = ?
y = x * 3
z = y - x
```



```
mov   r5, #3
mul   r2, r1, r5
sub   r3, r2, r1
cmp   r3, #4
beq   14 <success>
```

- Solve for x such that z == 4

Input

Constraint

Symbolic Execution Usages

- Reversing without reversing
  - Solve for input on stdin (flag) such that the flag checker prints “That flag is correct!”
- Automated PWN
  - Solve for input such that the instruction pointer is overwritten

Introducing Angr

- Angr can be used for automating CTF chals
- Install with `pip install angr`
- Template:
  - https://gist.github.com/richyliu/33489063d02c0a2afe0d6de6ec8d3e07

```python
import angr
import claripy

# replace with challenge name
project = angr.Project('./chal')

# tweak length if necessary
flag_len = 40
flag_chars = [claripy.BVS('flag_char_%d' % i, 8) for i in range(flag_len)]
# VERY IMPORTANT: add newline terminator if necessary (i.e. scanf)
symbolic_flag = claripy.Concat(*flag_chars + [claripy.BVV(b'\n')])

# can also pass in to argv
argv = [project.filename]
# unicorn for faster solve
state = project.factory.full_init_state(args=argv, add_options=angr.options.unicorn, stdin=symbolic_flag)
```

```python
for (i, flag_char) in enumerate(flag_chars):
    # tweak constraints if necessary
    char_constraint = claripy.And(flag_char >= ord('a'), flag_char <= ord('z'))
    char_constraint = claripy.Or(char_constraint, flag_char == ord('_'))
    # this is mostly likely needed
    char_constraint = claripy.Or(char_constraint, flag_char == 0x00)

    state.solver.add(char_constraint)

simgr = project.factory.simulation_manager(state)

print('exploring now...')
simgr.explore(
        # examples of correct and incorrect output
        find=lambda s: b'correct' in s.posix.dumps(1),
        avoid=lambda s: b'wrong' in s.posix.dumps(1))

# print flag once done
for found in simgr.found:
    print(found.solver.eval(symbolic_flag, cast_to=bytes))

print('done')
```

Side channels

Instruction Counting

- Given a flag as input, count how many instructions are executed
  - More instructions executed => flag is closer to being correct
  - Requires that program stops once part of the flag is incorrect
  - Order that flag is traversed

```
if (!strcmp(user_input, true_flag)) {
    puts("Correct!");
} else {
    puts("Wrong flag");
}
```

Instruction Counting

- Intel's Pin
  - https://github.com/ChrisTheCoolHut/PinCTF
- Can use valgrind's exp-bbv or callgrind tool
  - valgrind --tool=exp-bbv ./a.out sigpwny{...}
- aaaaaaa => 148862 instructions
- sigpwny => 148962 instructions
- Example of custom tool
  - https://gist.github.com/richyliu/468b926819b135a58a6936998f6100ca

Obfuscation

Self Modifying Code

- Code that modifies itself
- Use a debugger

```
_start:
 b8 3c 00 00 00
 mov     eax,0x3c
 b3 5b
 mov     bl,0x5b
 28 1d 05 00 00 00
 sub     BYTE PTR [rip+0x5],bl
 bf 00 00 00 00
 mov     edi,0x0
 6a 05
 push    0x5
 bf 02 00 00 00
 mov     edi,0x2
 0f 05
 syscall
```

```
0x4000d4 <_start>      mov    eax, 0x3c
0x4000d9 <_start+5>    mov    bl, 0x5b
► 0x4000db <_start+7>  sub    byte ptr [rip + 5], bl      <0x4000e6>
 ↓
0x4000db <_start+7>    sub    byte ptr [rip + 5], bl      <0x4000e6>
```

```
0x4000d4 <_start>      mov    eax, 0x3c
0x4000d9 <_start+5>    mov    bl, 0x5b
0x4000db <_start+7>    sub    byte ptr [rip + 5], bl      <0x4000e6>
► 0x4000e1 <_start+13> mov    edi, 1
0x4000e6 <_start+18>   syscall
0x4000e8 <_start+20>   mov    edi, 2
0x4000ed <_start+25>   syscall
```

Normally, program code is not modifiable.
Compile with `gcc -nostdlib -static -Wl,--omagic assembly.S -o bin` to make text segment writable.

VM Obfuscation

- Virtual machine executing other program instructions
  - Reasoning: lack of tools for custom VM
  - VMProtect, ropfuscated, hell
- Understand mode of instruction execution, writing tools (disassemblers, decompilers)
  - Find patterns
  - Work your way up the "abstraction ladder"

Go try for yourself!

- https://ctf.sigpwny.com
- Link again for angr solver script
  - https://gist.github.com/richyliu/33489063d02c0a2afe0d6de6ec8d3e07
- `pip install angr`
- `pip install z3-solver`
- Intel Pin (see README in downloaded zip file)

Next Meetings

2022-10-23 - This Sunday

- Guest Speaker: Mingjia
- "All Eyes On Me: Inside Third Party Trackers' Exfiltration of PHI from Healthcare Providers' Online Systems"

2022-10-27 - Next Thursday

- Social event TBD

2022-10-30 - Next Sunday

- Halloween Party 🎃

SIGPwny