Constraint Solvers for Reverse Engineering

Edgar Barbosa

H2HC 2018

Sao Paulo

Brazil

Disclaimer

- Opinions expressed are solely **my own** and do not express the views or opinions of my employer.

Objective of the talk

- Practical, **demonstration** based talk about how to integrate SMT solvers to your reverse engineering workflow.
- No academic theory, no discussions about SAT and SMT internals.
- Focus on demonstration of state-of-art SMT-based program analysis tools.

Constraint Solvers

- Constraint Programming
- Express relations between variables in the form of constraints
- Results:
  - sat – the constraints are satisfiable
  - unsat – it is not possible to satisfy the constraints
  - timeout – it can take a long time to find the answer (unknown)
  - crash (seriously) – how much RAM do you have?

Microsoft Z3

- Extremely powerful constraint solver (prover)
- Bindings to Python – easy to use
- Open-source
- Very popular SMT solver
- There are other solvers:
  - CVC4
  - STP
  - Yices

Language

- Do you like LISP?
    - You will love SMT-LIB!
- Official standard language for SMT solvers
- Documentation http://smtlib.cs.uiowa.edu/

DEMO

Workflow

- Has constraints to solve?
- Translate them to SMTLIB and use the power of the solvers
- Let’s do it!

DEMO

Making our life easier

- Some blessed soul at Microsoft decided to free us from the pain of using SMTLIB and create a Python bindings for the Z3 API.
- from z3 import *
- Theories (Integers, BitVectors, Reals, String, Sequences, RegEx)

DEMO

SMT and x86 code

- “Mr. Barbosa, I want to translate all world problems to SMT and retire! Teach me how do it!”
- Hold on! SMT solvers are really powerful, but P is still not equal NP, combinatorial explosion is still a thing and there is a lot of translation work to be done yet.
  - Example: translation of the x86 instruction set to SMT formulas.
  - Good news: very smart people has been working on building awesome tools to deal with x86 code. I will demo some of these tools.

Intermediate languages

- The reverse engineering equivalent of JavaScript frameworks.
- Makes sense, everyone has different objectives and preferences
- Some tools do not use an intermediate language
- We want to translate x86 to some IL. Some options:
  - OpenREIL
  - Valgrind - pyvex
  - Others (not enough time to demo them all)

DEMO

Tools

- Frequently one of the main objectives of using SMT based program analysis tools for x86 is to **automatically find** new inputs able to reach some execution path.

```
if (constraint1):
  if (constraint2):
    bug()
```

- Satisfy constraint1 and constraint2 and you'll be able to reach **bug()**

DEMO time!

- I selected a few tools to demo. There are more tools. Google is your friend (probably not). Links to tools in the Resources slide.
- Triton
- Manticore (Trail of Bits)
- Angr
- Klee
- mcsema (Trail of Bits)
- Ponce (IDA Pro plugin)

What’s next?

- Program analysis and SMT solvers are a hot topic now.
- Much easier to find online material now compared with some years ago.
- The are tons of examples of how to use SMT solvers to solve CTF challenges.
- A really nice are of research is AEG (make sure you don’t miss Thais Moreira’s presentation here at H2HC!!)
- Another one: Program Synthesis! https://homes.cs.washington.edu/~bornholt/post/building-synthesizer.html
- Topic for a future talk ☺

QUESTIONS?

Resources:

- Z3: https://github.com/Z3Prover/z3
- Manticore: https://github.com/trailofbits/manticore
- mcsema: https://github.com/trailofbits/mcsema
- angr: https://github.com/angr/angr
- Triton: https://github.com/JonathanSalwan/Triton
- pyvex: https://github.com/angr/pyvex
- klee: https://github.com/klee/klee
- ponce: https://github.com/illera88/Ponce