Application of Dynamic Symbolic Execution to Real-world Binary Programs

Duc Bui Hoang, Yunho Kim and Moonzoo Kim
ducbuihoang@kaist.ac.kr kimyunho@kaist.ac.kr moonzoo@cs.kaist.ac.kr
Provable Software Lab - KAIST

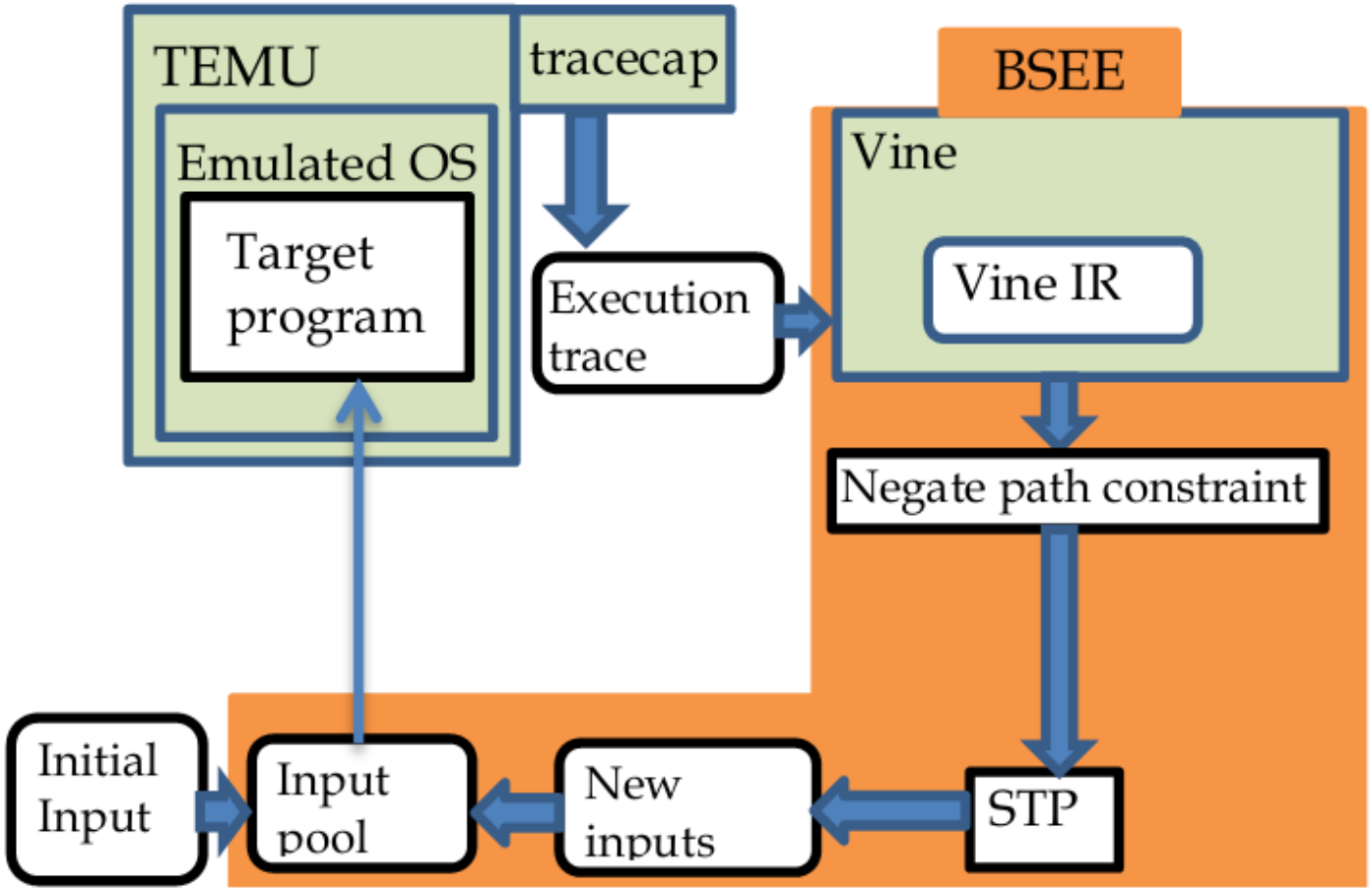
Motivations & Goal

- Analyzing binary programs is always in great demand.
- Dynamic symbolic execution (DSE) is a popular white-box testing technique.
  - Also called concolic testing

- Goal: Evaluation of the applicability of DSE technique to real-world Windows application programs
  - We applied the DSE technique to Notepad and Adobe Acrobat Reader on Windows XP and evaluated the experimental results.

Binary Symbolic Execution Engine (BSEE)

- BitBlaze is a binary analysis platform developed by Song et al. at UCB
- 2 main components of BitBlaze
  - TEMU: Dynamic analysis component.
  - Vine: Static analysis component.
- Our BSEE is built on top of Vine



Vine IR:Vine Intermediate Representation

BSEE architecture

Example

```
if(x!=5)
```

GNU C compiler

```
0x0804841b: cmp    $0x5, %eax
0x0804841e: je     0x804842c <main+56>
```

x=1 TEMU

Execution trace with x=1:

```
0804841b: cmp  $0x5,%eax
0804841e: je  0x0804842c
08048420:  … (not jump)
```

Vine

Vine IR:

```
/*cmp    $0x5,%eax*/
T_81_1520:reg32_t = R_EAX_5:reg32_t - 5:reg32_t;
R_ZF_13:reg1_t = T_81_1520:reg32_t == 0:reg32_t;
/*je     0x000000000804842c*/
cond_960:reg1_t = R_ZF_13:reg1_t == false;
assert(cond_960:reg1_t);
…
```

BSEE

Vine IR, after negation:

```
cond_960:reg1_t = R_ZF_13:reg1_t == false;
assert(!cond_960:reg1_t);
//End of file
```

STP

New input x=5:

0x5

Experiment Setting

Target programs	notepad.exe in Windows XP SP3 (5.1.2500.5512)
	AcroRd32.exe – The main executable file of Adobe Acrobat Reader 9.2.0

- Symbolic input is a file
  - A 743-byte long file consisting of character '0'
  - 737 bytes is the minimal size which enables instruction tracing function. A 743-byte long file size is needed to generate a new input of 737 bytes long. (BitBlaze's limitation).

Results

- Adobe Acrobat Reader: 1469 test cases were generated in more than 5 hours

Run number of BSEE	Initial input	Number of new inputs	Lengths of new inputs	Tracing time	Test case generation time
1	743 bytes of character ‘0’	736	1 to 738 bytes	10 min	2 hours
2	The 738-byte long input in the first run	733	1 to 733 bytes	10 min	2.5 hours

- Notepad: we could not generate test cases because Vine encountered an error when translating the execution trace to Vine IR

Lessons learned (1/2)

- Limitations of BitBlaze
  - TEMU could not record the execution trace when the target program reads a very small file.
  - TEMU can miss propagation of tainted data in the executions of complicated applications.
  - Vine failed to handle certain binary instructions.
  - Analysis of BitBlaze is too slow and consumes too much of resources (see the table below).

BitBlaze Performance

Target program	AcroRd32.exe			notepad.exe
Tracing time	10min	15min	60min	1min
Size of trace file	1.2GB	2.1GB	35.0GB	72MB
Translation time (execution trace to Vine IR)	2min	out of memory	out of memory	1min (interrupted by an error)
Size of Vine IR	23MB	N/A	N/A	N/A

Lessons learned (2/2)

- Large amount of low level data
  - 10-minute execution trace of Acrobat Reader is 1.2GB and contains more than 19 million instructions.
  - We need to process executed instructions of external libraries and environment in addition to the executed instructions of the target program.
    - Separate instructions executed by the target program from external libraries and the operating system.
  - This problem reduces the scalability of the tool

Summary

- We applied a BitBlaze-based symbolic execution engine to 2 real-world application programs on Windows.
- As a result, we could generate hundreds of test cases for Acrobat Reader while we could not generate test cases for Notepad on Windows XP.
- We found that there are still many challenges and limitations of the existing tool that make DSE not applicable to real-world applications at the operating system level.


Thank you
...
Questions?