Find your Path – SOK Path exploration (in progress)

Dairo de Ruck





- A module of the symbolic engine

> Algorithmic choice



- > Algorithmic choice
 - >> Next exploration step



- > Algorithmic choice
 - » Next exploration step

>> (sometimes) context-aware



- > Algorithmic choice
 - » Next exploration step

» (sometimes) context-aware

» Significant impact on global efficiency



- Algorithmic choice >
 - » Next e

- » (sometimes) context-aware
- sermon/ » Significant impact on global efficiency



Path Exploration – working example

How to detect bug in the red
Basic Block

> Exhaustive vs Efficient





Looking at a real Control Flow
Graph

 This choice has a major impact



9

CFG was too big for PowerPoint to handle... Use your imagination ©



STA

> "Optimal" symbolic execution:

» Explore all possible paths

>> Find EVERY vulnerability



> "Optimal" symbolic execution:

» Explore all possible paths

» Find EVERY vulnerability



- > ... An Optimal world does not exist:
 - » Memory is finite
 - » Time is money
 - » World peace is yet to be achieved



> ... An Optimal world does not exist, so:

» Find as many "targets" as possible within a reasonable time



> ... An Optimal world does not exist, so:

>> Find as much "targets" as possible within a reasonable time





SOK





- > Various characteristics defined
 - » Overarching selection strategies
 - » Use of Search Space Optimizations (SSO)

$ \begin{bmatrix} III \\ IIII \\ O, D, D \\ IIIII \\ IIII \\ IIIIII$		Goal	Selection Strategy			SSO			ar	oť	뎡
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$		Targeted/Overall	Naive/Heuristic	Learning based	Knowledge based	State merging	Path Pruning	Constraint reasoning	Ye	Extension	Operates
[63]* T x x '09 .NET [68] T x x '10 KLEE LLVM IR [21] O x x '11 angr VEX IR [39]* T H '11 SAGE binary [36]* O x x '12 KLEE LLVM IR [18] O H '12 AEG binary [37]* O x x '13 KLEE LLVM IR [14] O x x '14 LLVM IR [14] O x x '14 P [47]* O ? x x x'16 KLEE LLVM IR [43]* T x x x'16 KLEE LLVM IR [60] O I -'20 AST [43]* [60] O X x '20 SAGE binary	[17]	0	N+H						'08		LLVM IR
[68] T x x '10 KLEE LLVM IR [21] O x x '11 angr VEX IR [39]* T H '11 SAGE binary [36]* O x x '12 KLEE LLVM IR [18] O H '12 KLEE LLVM IR [13] T N x '13 KLEE LLVM IR [13] T H '14 LLVM IR R [14] O x x '14 LVM IR [14] O x x '16 KLEE LLVM IR [47]* O ? x x'15 KLEE LLVM IR [42] T x x '16 KLEE LLVM IR [43]* O x x '20 AST [60] O I '20 AST [61]	[63]*	Т			x				'09		.NET
[21] O x x '11 angr VEX IR [39]* T H '11 SAGE binary [36]* O x x '12 KLEE LLVM IR [18] O H '12 AEG binary [37]* O x '13 KLEE LLVM IR [13] T H '14 LLVM IR [13] T H '14 LLVM IR [14] O x x' '14 LLVM IR [13] T H '14 LLVM IR [14] O x x' '15 KLEE LLVM IR [14] O x x x'16 KLEE LLVM IR [42]* T x x '20 AST [43]* O x x'20 SAGE binary [16] X X '20 KLEE LLVM	[68]	Т			x				'10	KLEE	LLVM IR
[39]* T II II SAGE binary [36]* O x x x '11 SAGE binary [18] O H '12 KLEE LLVM IR [18] O H '12 AEG binary [37]* O x '13 KLEE LLVM IR [13] T H '14 LLVM IR [14] O x x '14 LLVM IR [14] O x x '16 KLEE LLVM IR [47]* O ? x x'16 KLEE LLVM IR [42] T x x x'16 KLEE LLVM IR [42] T x x '20 AST [60] O H '20 AST [61] O x '20 KLEE LLVM IR [62]* X '20 KLEE LLVM IR Image [65] T x '21 angr	[21]	0		x	x				'11	angr	VEX IR
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[39]*	Т	Н						'11	SAGE	binary
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[36]*	0			x	x			'12	KLEE	LLVM IR
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[18]	0	Н						'12	AEG	binary
[20]* T N x x x '13 KLEE LLVM IR $[13]$ T H '14 LLVM IR '14 LLVM IR $[14]$ O x x '14 LLVM IR $[14]$ O x x '14 LLVM IR $[14]$ O ? x x' '15 KLEE LLVM IR $[28]*$ T x x x' 16 KLEE LLVM IR $[42]$ T x x x' 16 KLEE LLVM IR $[42]$ T x x '20 SAGE binary $[60]$ H - '20 SAGE binary $[16]$ x x '20 KLEE LLVM IR $[62]*$ O x - '21 angr VEX IR $[65]$ T x - '21 angr VEX IR	[37]*	0			x				'13	KLEE	LLVM IR
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	[20]*	Т	N		x		x		'13	KLEE	LLVM IR
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[13]	Т	Н						'14		LLVM IR
	[14]	0			x		x		'14		?
$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	[47]*	0	?				x	x	'15	KLEE	LLVM IR
[42] T x x x '16 KLEE LLVM IR $[60]$ 0 X '19 angr VEX IR $[60]$ 0 H '20 AST $[43]$ * 0 X X '20 SAGE $[16]$ 0 X X '20 SAGE $[16]$ 0 X X '20 KLEE $[16]$ 0 X X '20 KLEE LLVM IR $[62]$ * 0 X -'21 KLEE LLVM IR $[30]$ * 0 X -'21 angr VEX IR $[65]$ T X -'21 angr VEX IR $[66]$ T X -'21 angr VEX IR $[66]$ T X -'21 angr VEX IR $[56]$ O X X '22 binary $[41]$ * O X X	[28]*	Т			x		x	x	'16	KLEE	LLVM IR
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[42]	Т			x			x	'16	KLEE	LLVM IR
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[69]*	0			x				'19	angr	VEX IR
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[60]	0	Н						'20		AST
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[43]*	0		x			x		'20	SAGE	binary
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$	[16]	0			x			x	'20	KLEE	LLVM IR
[30]* O x '21 KLEE LLVM IR [48]* O x '21 angr VEX IR [65] T x '21 angr VEX IR [66] T x '21 angr VEX IR [66] T x '21 angr VEX IR [26]* O x '21 spF Java bitcode [41]* O x x '22 binary [56] O x x '22 binary [45] T H '23 binary [31] T x x '23 binary [67] T N x '23 binary [49] O X '23 Cover VEX IR [32] O X '23 Cover VEX IR [32] O X '23 Cover VEX IR	[62]*	0		x					'20	KLEE	LLVM IR
[48]* O x '21 angr VEX IR [65] T x '21 angr VEX IR [66] T x '21 angr VEX IR [66] T x '21 angr VEX IR [26]* O x '21 SPF Java bitcode [41]* O x x '22 Cover VEX IR [56] O x x '22 binary [45] T H '23 binary [31] T x x '23 binary [67] T N x '23 binary [49] O x '23 Cover VEX IR [32] O x '23 binary [49] N x '23 Cover VEX IR [32] O X '23 Cover VEX IR [[30]*	0		x					'21	KLEE	LLVM IR
Image: Image in the i	[48]*	0		x					'21	angr	VEX IR
i66 T x '21 angr VEX IR [26]* O x '21 SPF Java bitcode [41]* O x x '21 SPF Java bitcode [41]* O x x x '22 Cover VEX IR [56] O x x '22 binary [45] T H '23 binary [31] T x x '23 binary [51] O	[65]	Т			x				'21	angr	VEX IR
[26]* O x x '21 SPF Java bitcode [41]* O x x x '22 Cover VEX IR [56] O x x x '22 binary [41]* O x x '22 binary [45] T H '23 binary [45] T H '23 webASM [51] O '23 binary [67] T N x x '23 binary [49] O x '23 Cover VEX IR [54]* O N x '23 Cover VEX IR [32] O x '23 SPF Java bitcode [22]* O H '23 SPF Java bitcode	[66]	Т			x				'21	angr	VEX IR
[41]* O x x x x '22 Cover VEX IR [56] O x x x '22 binary [45] T H '23 binary [31] T x x '23 binary [51] O - '23 binary [67] T N x x '23 binary [67] T N - '23 c-prog [54]* O X x '23 C-prog [53]* O N x x '23 SPF [32] O X '23 SPF Java bitcode [22]* O H '23 binary	[26]*	0			x				'21	SPF	Java bitcode
156 O x x 22 binary [45] T H '23 binary [31] T x x '23 binary [31] T x x '23 binary [51] O '23 binary [67] T N x x'23 binary [49] O x '23 C-prog [54]* O N x x'23 Cover [52] O N x x'23 Cover VEX IR [32] O x '23 SPF Java bitcode [22]* O II '23 binary	[41]*	0		x	x			x	'22	Cover	VEX IR
[45] T II × × 23 binary [31] T x x ×23 WebASM [51] O - '23 binary [67] T N x x'23 binary [67] T N x x'23 binary [49] O - x '23 Coreg [51] O - x '23 binary [67] T N x x'23 binary [49] O - x '23 Coreg [51] O N x x '23 Coreg [52] O I x '23 SPF Java bicode [22]* O II '23 binary	[56]	0			x			x	'22		binary
Image: Text of the system x x x '23 WebASM [51] O	[45]	Т	Н						'23		binary
[51] O '23 binary [67] T N x x'23 binary [49] O x '23 Cover C-prog [54]* O N x x'23 Cover VEX IR [32] O X x'23 Cover VEX IR [22]* O II '23 SPF Java bitcode	[31]	Т			x		x		'23		WebASM
[67] T N x x '23 binary [49] O x '23 C-prog [54]* O N x x '23 Cover VEX IR [32] O x '23 SPF Java bitcode [22]* O H '23 binary	[51]	0							'23		binary
[49] O x '23 C-prog [54]* O N x x '23 Cover [32] O X x '23 SPF Java bitcode [22]* O H '23 binary	[67]	Т	N				x	x	'23		binary
[54]* O N x x x '23 Cover VEX IR [32] O x '23 SPF Java bitcode [22]* O H '23 binary	[49]	0			X				'23		C-prog
[32] O x '23 SPF Java bitcode [22]* O H '23 binary	[54]*	0	N	x	x			x	'23	Cover	VEX IR
[22]* O H '23 binary	[32]	0				x			'23	SPF	Java bitcode
	[22]*	0	Н						'23		binary

> How to execute path exploration in real life?



> How to execute path exploration in real life?

» "Blind"

» Informed (in any way)





> How to execute path exploration in real life?





> Relate to the SUT (subject under test)

> .. or to the algorithm in use



- > Relate to the SUT
 - » (CFG, node connectivity, loops, branch hit probabilities,...)

- > .. or to the algorithm in use
 - » (DFS, BFS, RNG element, lowest path costs, ...)



- > More advanced:
 - » Potential presence of bugs



- > More advanced:
 - » Potential presence of bugs
 - » Certain presence of bugs



- > More advanced:
 - » Potential presence of bugs
 - » Certain presence of bugs
 - » Insertion of domain knowledge



- > More advanced:
 - » Potential presence of bugs
 - » Certain presence of bugs
 - » Insertion of domain knowledge
 - » Preprocessing SAST/DAST step

>>





CFG spectra

Distance

- Constraints
- Potential bugs
- Certain presence of bugs
- Critical operations
- Domain knowledge
- Prune set

Branch hit probabilities

Path Exploration - Techniques

- > "Blind" search
- > Informed search
 - » Knowledge based search

>>>

» Learning based search

>>>



Path Exploration - Techniques

> "Blind" search

- >> DFS
- » BFS
- » Random (state/path) Search
- > Informed search
 - » Knowledge based search

»»

» Learning based search

>>>





Path Exploration - Techniques



> "Blind" heuristics

- ^{>>} More general → less efficient
- » BUT predictable memory usage
- > Current Learning based approaches
 - » (Over)engineered to a single set of "vulnerabilities"
 - » No sense of global landscape
- > Use of other external information
 - >> Point to specific locations
- > Hybrid approaches (Fuzzer/SymEx or SAST/SymEx)
 - >> Very efficient, as they can complement each others' weaknesses

 Categorization of the path exploration techniques

Categorization of features
used (f. ex., pie chart)



- > No uniform way to compare techniques...
 - » angr vs KLEE vs new implementation
 - >> Targeted vs global
 - ›› ... VS ...
- No uniform benchmark…
 - » GNU Core utils, Juliet, RW bins, LAVA bench, SPEC CPU,

- > No uniform metric
 - >> Coverage, vulnerabilities detected, accuracy, F1, vulnerability types,





- 2 options
- > Reimplement the more prominent techniques

OR

> Create a uniform way to compare engines/techniques



2 options

Reimplement the more prominent techniques

OR

> Create a uniform way to compare engines/techniques





> Comparative study of the exploration techniques

- » Reimplemented in the same engine
- » On the same set of binary





> Comparative study of the exploration techniques

» Reimplemented in the same engine

>>> TBD

» On the same set of binary

» CGC, CORE Utils, JULIET, ...



Where do you think the added value lies in a benchmark?

Should there be more guidelines w.r.t. reproducibility?

Other caveats w.r.t. benchmarking these solutions?

DISTRINET Thank you!

Dairo.deruck@kuleuven.be

https://distrinet.cs.kuleuven.be/