

Satisfiability Modulo Theories and Network Verification

Nikolaj Bjørner
Microsoft Research
Formal Methods and Networks Summer School
Ithaca, June 10-14 2013

Lectures

Wednesday 2:00pm-2:45pm:

An Introduction to SMT with Z3

Thursday 11:00am-11:45am

Algorithmic principles of SAT/SMT

Friday 9:00am-9:45am

Theories, Solvers and **Applications**

Z3 source is (evil) subverting 8/



Google

שתף את הדברים הנכונים בדיוק עם האנשים הנכונים.

הצטרף ל-Google+



גר ב- Munich, Germany

הצג את הפרופיל המלא

Jan Wildeboer



Jan Wildeboer 3 באוק 2012 - גלוי לכולם

Microsoft trying to subvert Open Source. Not the first time: <http://research.microsoft.com/en-us/um/people/leonardo/blog/2012/10/02/open-z3.html>

"Z3 is now open source. The source code is available under the MSR-LA license. This is the same license used to distribute the Z3 binaries. The source code can be used for non-commercial purposes. "

Note the "non-commercial". This is not open source with such a limitation. It fits the political agenda of Microsoft however, where they have told politicians again and again the they think Open Source means non-commercial. But it simply isn't.

Open Source means Open Source. And that means no limitations on use and reuse.

« Open Sourcing Z3

Leonardo de Moura' Homepage



File Edit View Help

resurrecting ass...
 fix if lifting, addc...
 Merge branch 'ur...
 old_params ==>...
 Fixed bug reported by Arie Gu...
 moved smt tactic to smt folder
 Moved dead code to dead branch
 optimizing DL
 disable buggy code in slicer: it removes conjuncts for non-sliced variables.
 removed dead code
 fixed typo
 Merge branch 'unstable' of https://git01.codeplex.com/z3 into unstable
 Merge branch 'unstable' of https://git01.codeplex.com/z3 into unstable
 Merge branch 'unstable' of https://git01.codeplex.com/z3 into unstable
 fix bugs in inliner and usage of unbound variable fix, reported by Arie Gu...
 Merge branch 'unstable' of https://git01.codeplex.com/z3 into unstable
 local changes
 removing fat
 bindings -> api; and moved nlsat/sat/subpaving tactics
 added add_extra_exe command to build framework
 enable pdb for release mode 32bit
 add default template instance
 missing update
 more cleanup
 make front_end_params an optional argument in cmd_context
 Fixed warnings reported by gcc 4.7.1
 Fixed warnings reported by gcc 4.7.1
 Merge branch 'unstable' of https://git01.codeplex.com/z3 into unstable
 added --nodotnet option to mk_make.py
 Fixed warnings produced by gcc 4.6.3 when compiling in debug mode
 Merge branch 'unstable' of https://git01.codeplex.com/z3 into unstable
 fix bugs encountered by regression tests
 Added LICENSE.txt to win bin distrib
 change share library search in Z3Py
 include VS redistributables in the win dist
 updated README

Z3 is open shared source

<http://z3.codeplex.com/>

Leonardo de Moura <leonardo@microsoft.com>	2012-11-01 21:15:45
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-11-04 08:40:16
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-11-04 22:06:16
Leonardo de Moura <leonardo@microsoft.com>	2012-11-01 20:28:14
Leonardo de Moura <leonardo@microsoft.com>	2012-11-01 19:28:26
Leonardo de Moura <leonardo@microsoft.com>	2012-11-01 17:48:54
Leonardo de Moura <leonardo@microsoft.com>	2012-11-01 17:40:20
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-11-01 22:06:10
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-11-01 05:29:28
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 23:58:21
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 23:36:18
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 23:22:00
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-10-31 22:35:45
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-10-31 22:25:42
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-10-31 22:23:24
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-10-31 19:37:10
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-10-31 19:37:05
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 23:21:22
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 22:24:39
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 22:14:37
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 22:09:05
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 20:16:43
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 20:02:14
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 19:54:59
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 18:43:46
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 09:16:26
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 09:05:38
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 08:48:23
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 02:47:37
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 08:43:00
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 01:42:36
Nikolaj Bjorner <nbjorner@microsoft.com>	2012-10-31 01:13:27
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 01:42:05
Leonardo de Moura <leonardo@microsoft.com>	2012-10-31 00:09:12
Leonardo de Moura <leonardo@microsoft.com>	2012-10-30 23:17:02
Leonardo de Moura <leonardo@microsoft.com>	2012-10-30 20:20:07

SHA1 ID: f44631ce73e9e1d1f97416ca08dbb3618fc930cb

Row 53 / 375

Find next prev commit containing:

Search

Diff Old version New version Lines of context: 3 Ignore space change Line diff

author: Nikolaj Bjorner <nbjorner@microsoft.com> 2012-10-31 01:13:27

Committer: Nikolaj Bjorner <nbjorner@microsoft.com> 2012-10-31 01:13:27

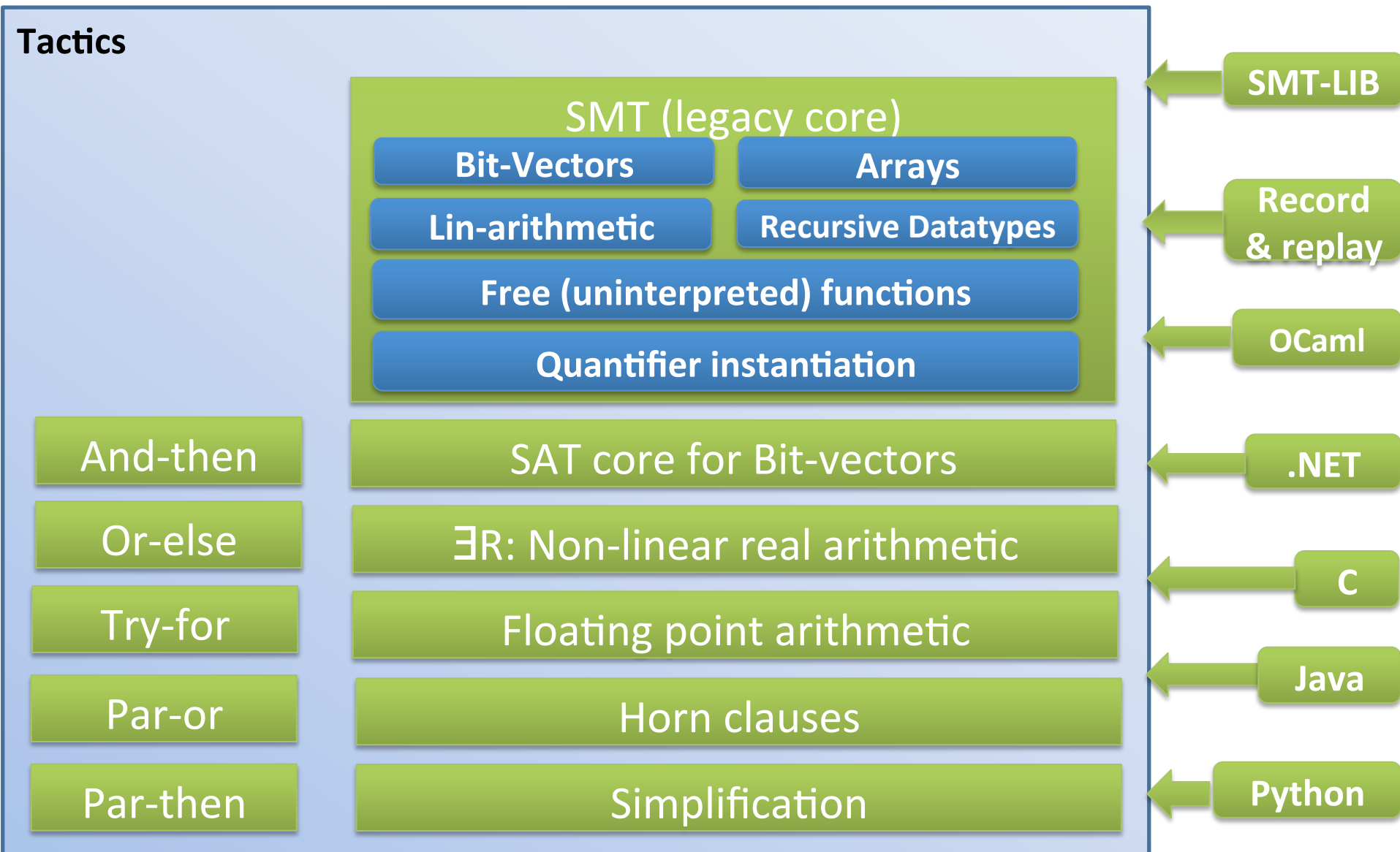
parent: ec907a470518bf9e46f2ddd18ea1b1161db74dda (change share library search in Z3Py)

Patch Tree

Comments

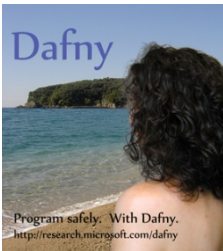
src/ast/ast.cpp
 src/muz_qe/dl_util.cpp
 src/muz_qe/pdr_context.cpp

Z3 architecture - new



Some Microsoft Tools based on

Z3



Program
Verification



HAVOC

Auditing



Type Safety

SLayer



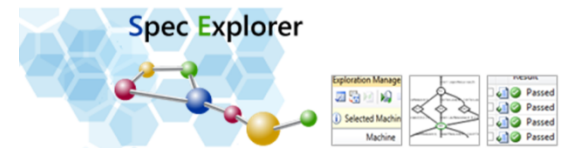
Over-
Approximation

TERMINATOR



Under-
Approximation

SAGE

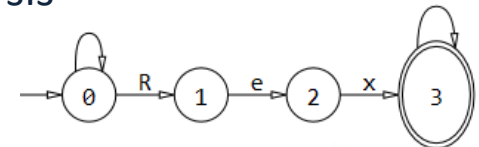


Testing

M3



Analysis



Synthesis



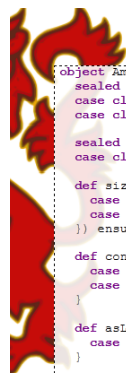
Cool tools using **Z3**

Sledge
Hammer



Liquid Types

LeonOnline



up at
copy-
in this

```
object AmortizedQueue {
  sealed abstract class List
  case class Cons(head : Int, tail : List) extends List
  case class Nil() extends List

  sealed abstract class AbsQueue
  case class Queue(front : List, rear : List) extends AbsQueue

  def size(list : List) : Int = (list match {
    case Nil() => 0
    case Cons(_, xs) => 1 + size(xs)
  }) ensuring(_ >= 0)

  def content(l: List) : Set[Int] = l match {
    case Nil() => Set.empty[Int]
    case Cons(x, xs) => Set(x) ++ content(xs)
  }

  def asList(queue : AbsQueue) : List = queue match {
    case Queue(front, rear) => concat(front, reverse(rear))
  }

  def concat(l1 : List, l2 : List) : List = (l1 match {
    case Nil() => l2
    case Cons(x, xs) => Cons(x, concat(xs, l2))
  }) ensuring (res => size(res) == size(l1) + size(l2) && content(res) == content(l1) ++ content(l2))

  def isAmortized(queue : AbsQueue) : Boolean = queue match {
    case Queue(front, rear) => size(front) >= size(rear)
  }
}
```

Verify!

concat	postcond.	valid	Z3-f	0.011
reverse	postcond.	valid	Z3-f	0.028
amortizedQueue	postcond.	valid	Z3-f	0.004
enqueue	postcond.	valid	Z3-f	0.003
isAmortized	postcond.	valid	Z3-f	0.003



PUG

ESBMC

Scala^{Z3}

MetiTarski

KeYmaera



Jeves

Testing

Hunting for Security Bugs



- Two main techniques used by “black hats”:
 - *Code inspection* (of binaries) and *Blackbox fuzz testing*
- **Blackbox fuzz testing:**
 - A form of blackbox random testing
 - Randomly *fuzz* (=modify) a well-formed input
 - Grammar-based fuzzing: rules that encode how to fuzz
- **Heavily used in security testing**
 - At MS: various internal tools
 - Conceptually simple yet effective in practice...
 - Has been instrumental in weeding out 1000's of bugs during development and test

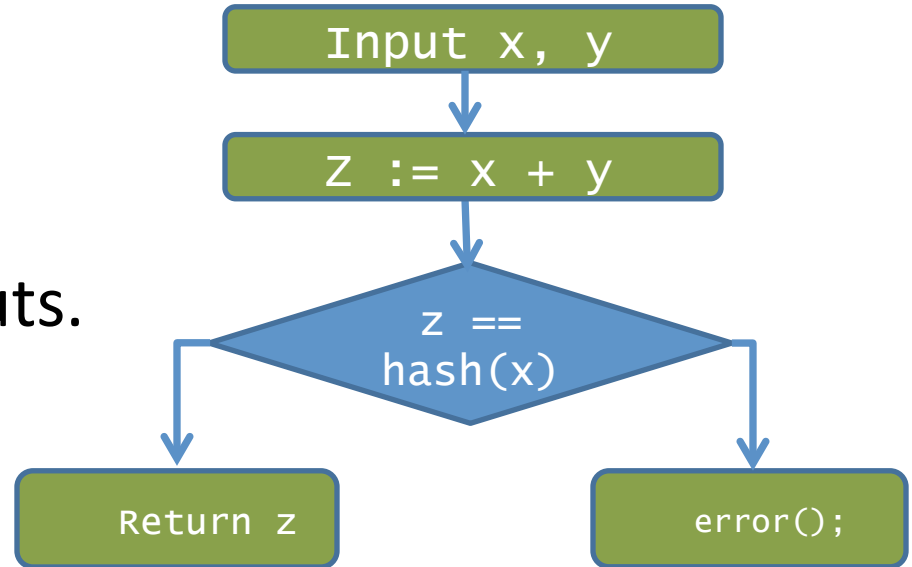
Method: Dynamic Test Generation

Run program with **random** inputs.

Gather constraints on inputs.

Use constraint solver to generate new inputs.

Combination with randomization: DART
Godefroid-Klarlund-Sen-05,...



Can't statically generate value for x that satisfy " $z == \text{hash}(x)$ "

But we can solve for y in:

$$x + y = \text{hash}(x)$$

Fuzzing and Test Case Generation

SAGE



Internal. For Security Fuzzing External. For Developers

Runs on x86 instructions

Runs on .NET code

Try it on: <http://pex4fun.com>

Finding security bugs before the hackers



black hat

Fuzzing and Test Case Generation

SAGE

Internal. For Security

Runs on x86 instructions

Dr. Strangelove?

*Bug: ***433*

*“2/29/2012 3:41 PM Edited by ******

SubStatus -> Local Fix

*I think the fuzzers are starting to become sentient.
We must crush them before it is too late.*

*In this case, the fuzzer figured out that if
[X was between A and B then Y would get
set to Z triggering U and V to happen.....]*

.....

***And if this fuzzer asks for the nuclear launch
codes, don't tell it what they are ...”***

Finding security bugs before the hackers



black hat

SAGE by numbers

100s CPU-years - largest dedicated fuzz lab in the world

100s apps - fuzzed using SAGE

100s previously unknown bugs found

Billion+ computers updated with bug fixes

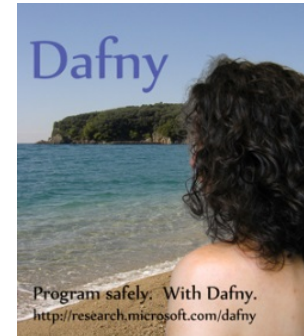
Millions of \$ saved for Users and Microsoft

10s of related tools (incl. Pex), 100s DART citations

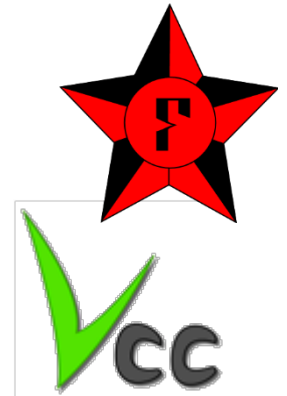
3+ Billion constraints - largest usage for any SMT solver



Verification
It is cool
easy
fun
attractive



Program
Verification



HAVOC

Auditing



Type Safety

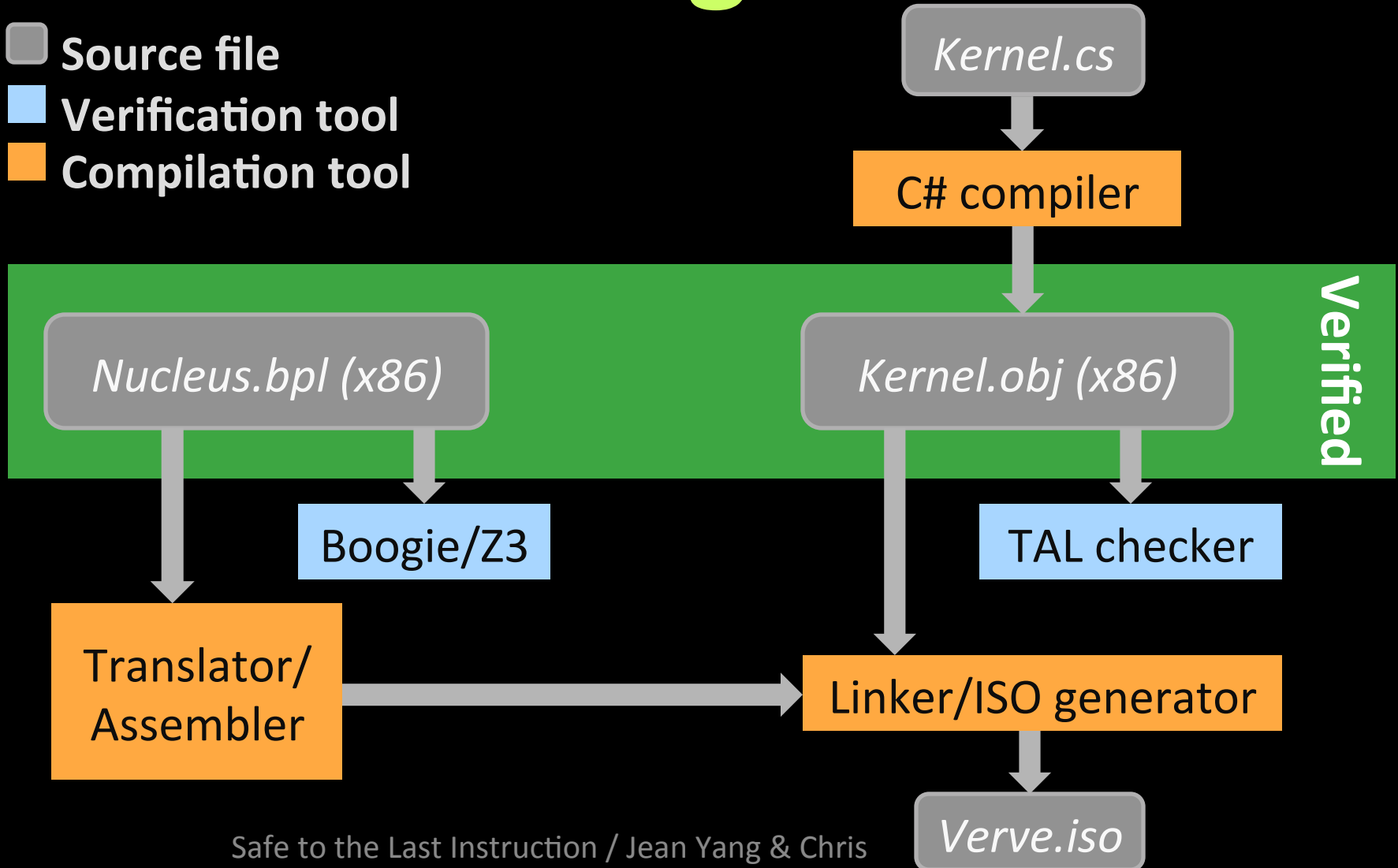
SLayer

Building Verve



9 person-months

- Source file
- Verification tool
- Compilation tool



Validation

SecGuru: Automatic Validation of Network Connectivity Restrictions



Microsoft
Research

Karthick Jayaraman, Charlie Kaufman,
and Ramanathan Venkatapathy

Nikolaj Bjørner

Network Policies:

Complexity, Challenge and Opportunity

Several devices, vendors, formats

- Net filters
- Firewalls
- Routers

Challenge in the field

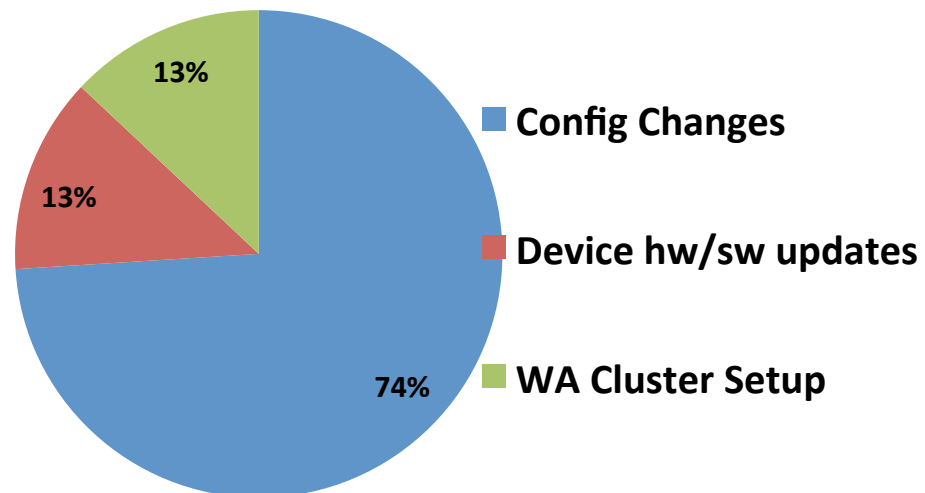
- Do devices enforce policy?
- Ripple effect of policy changes

Arcane

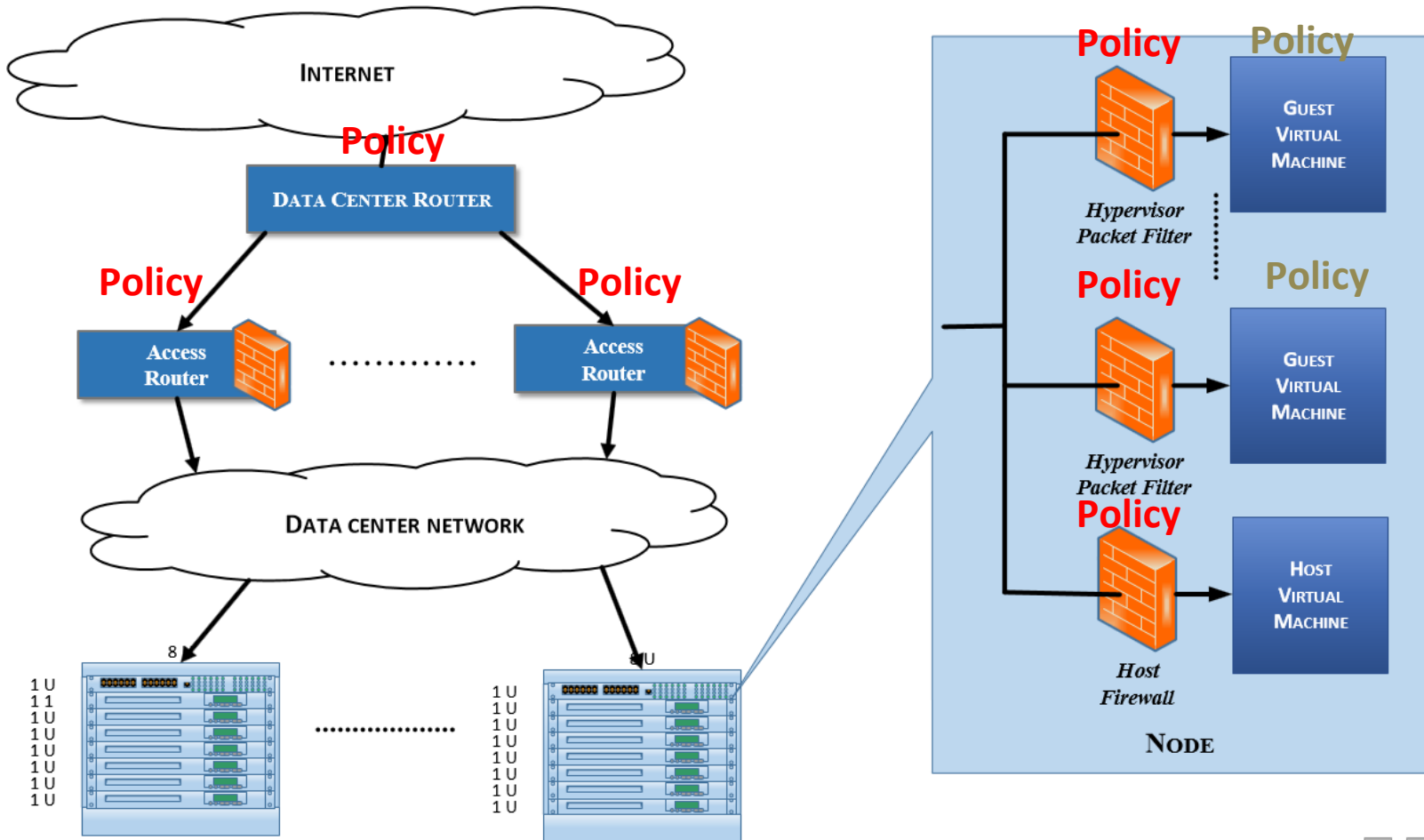
- Low-level configuration files
- Mostly manual effort
- Kept working by
“Masters of Complexity”

Human errors > 4 x DOS attacks

Human Errors by Activity



A Data-center Architecture



A Data-center Architecture

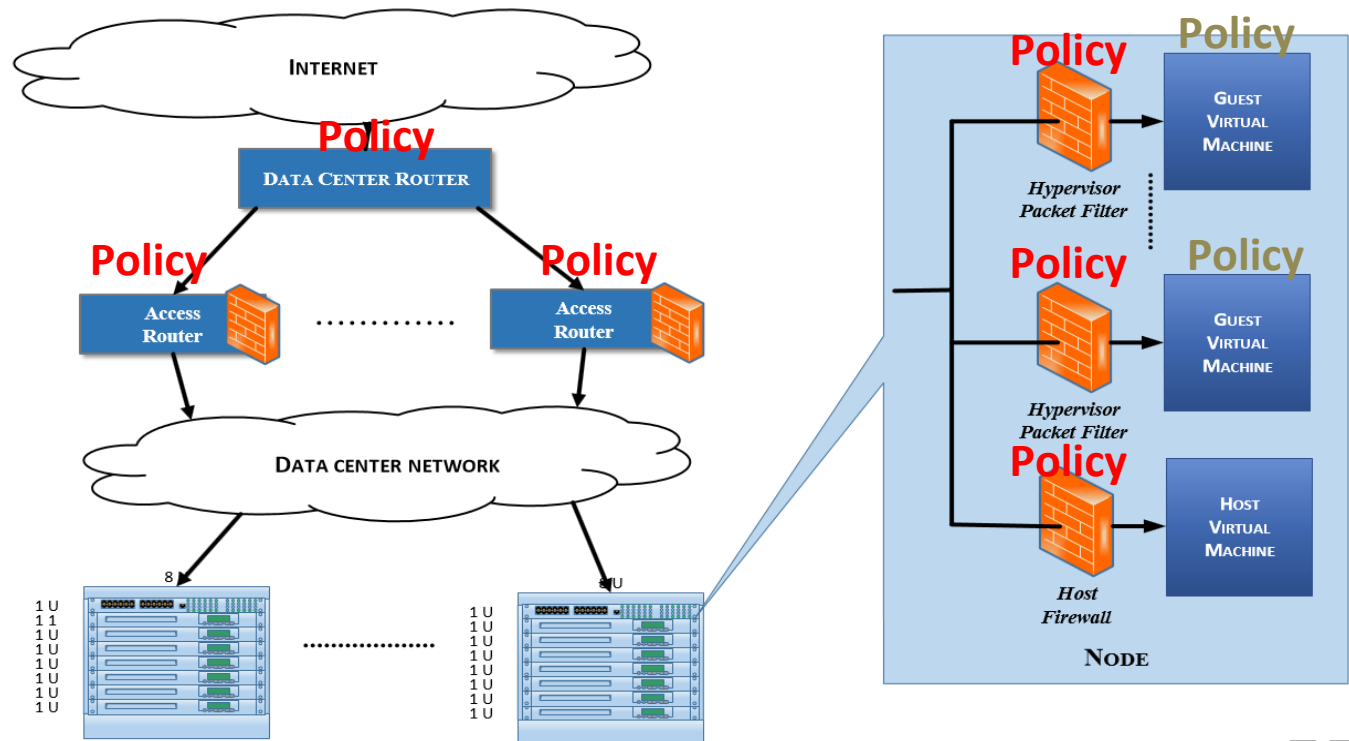
Defense in Depth = Crypto + Policies

- outside IP can be spoofed

Efficient and Flexible Defense by Policy only

- inside IP cannot be spoofed

Policies (Access Control Lists) matter



Network Policies

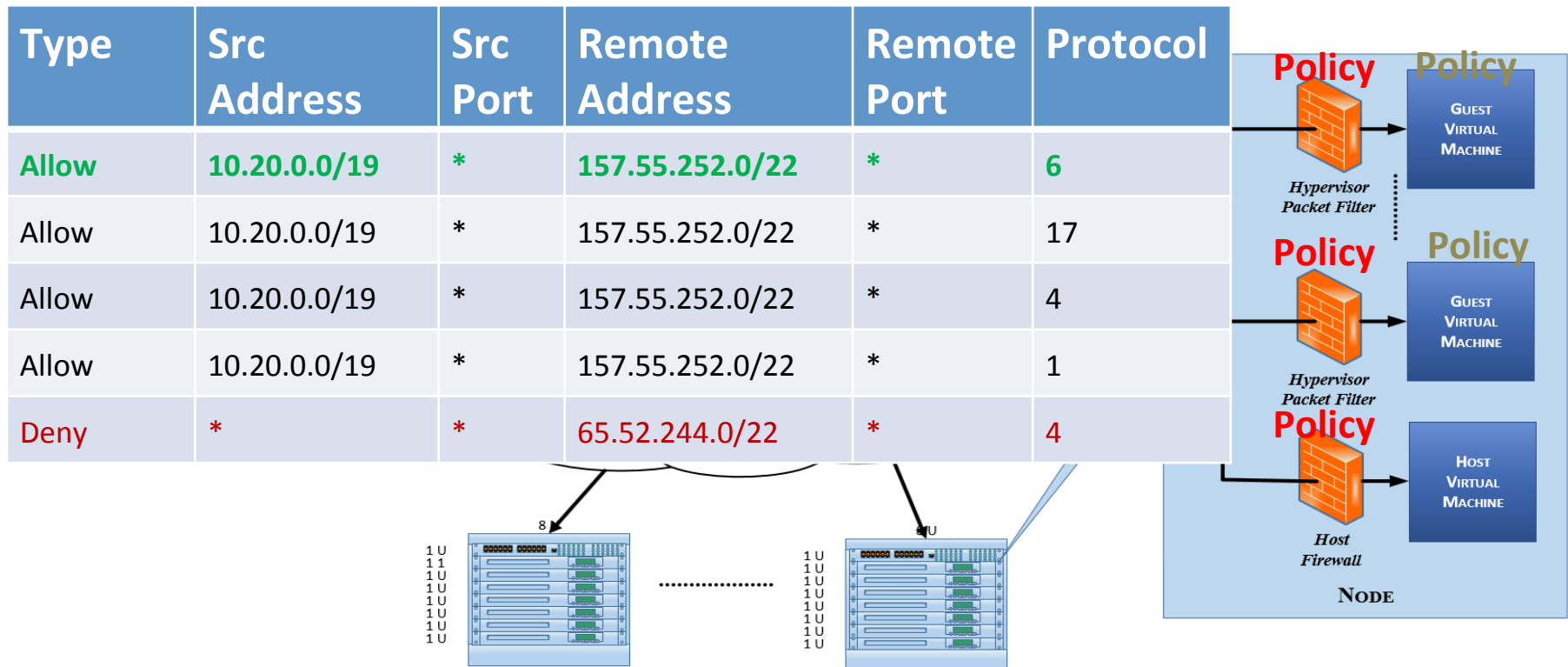
What they look like in routers

Defense in Depth = Crypto + Policies

- outside IP can be spoofed

Efficient and Flexible Defense by Policy only

- inside IP cannot be spoofed



A Need for Automating Policy Configuration

Constantly growing

New clusters – “same” policy, but different addresses

Constantly changing

Hardware – capacity, semantics

New Services – selectively exposed

Patches – to *live site incidents*

Bare metal low-level format of routers are also used for policy configurations

Towards automation: Checking Policy Configuration with **SecGuru**

Constantly growing

*New clusters – enforce and check that
new policies are instances
of a master template (the intent)*

Constantly changing

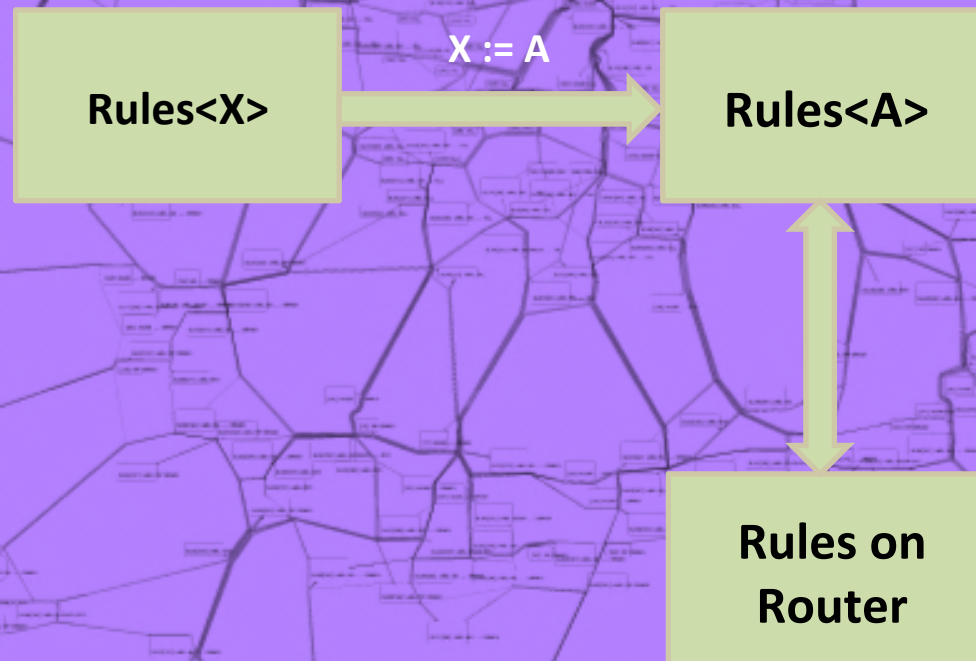
Hardware – capacity, semantics

New Services – check effect of new rules

Patches – check for regressions

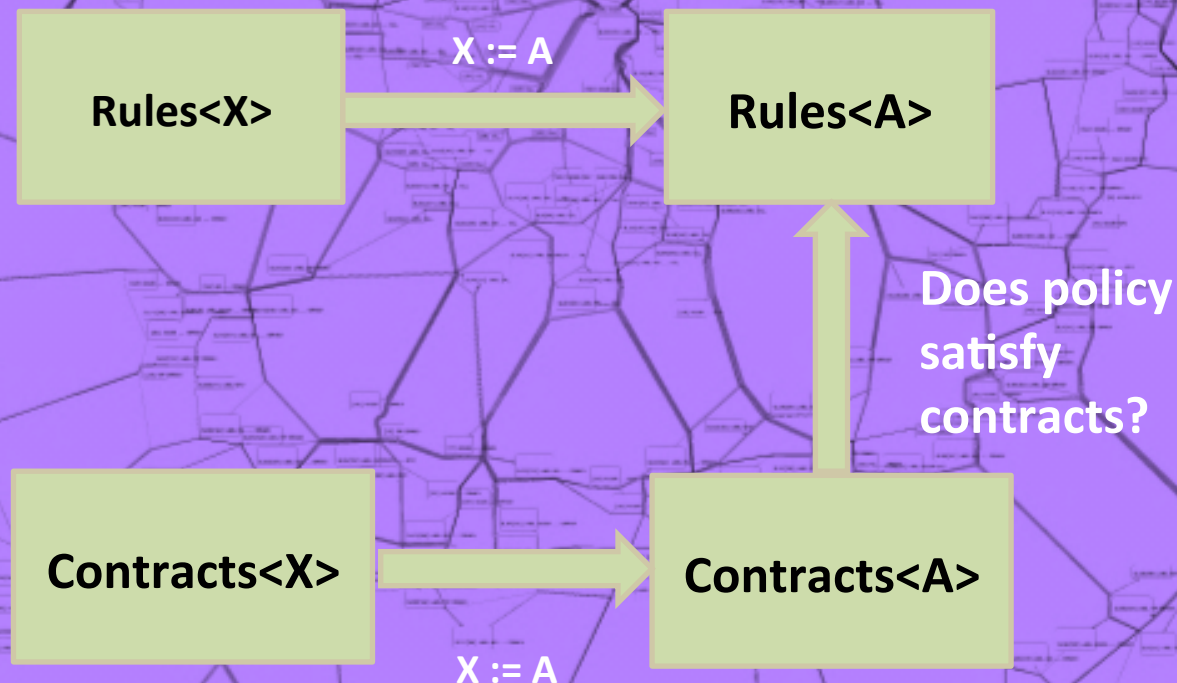
Towards automation: Checking Policy Configuration with **SecGuru**

Enforce and check that policies are instances of a master template (the intent)



Towards automation: Checking Policy Configuration with **SecGuru**

Enforce and check that policies satisfy contracts



Policies as Bit-Vector Formulas

Type	Src Address	Src Port	Remote Address	Remote Port	Protocol
Allow	10.20.0.0/19	*	157.55.252.0/22	*	6
Allow	10.20.0.0/19	*	157.55.252.0/22	*	17
Allow	10.20.0.0/19	*	157.55.252.0/22	*	4
Allow	10.20.0.0/19	*	157.55.252.0/22	*	1
Deny	*	*	65.52.244.0/22	*	4

IP, Port, and Protocol: **bit vectors**

Policy: **Bit-vector logic**

Allow: $(10.20.0.0 \leq s$
 $rclp\ 10.20.31.255)$
 $\wedge (157.55.252.0 \leq dstlp$
 $< 157.55.252.255) \wedge$

Policy: $(Vi \uparrow \text{Allow}$
 $w \downarrow i) \wedge (\wedge j \uparrow \neg Den$

Policies as Bit-Vector Formulas

Type	Src Address	Src Port	Remote Address	Remote Port	Protocol
Allow	10.20.0.0/19	*	157.55.252.0/22	*	6
Allow	10.20.0.0/19	*	157.55.252.0/22	*	17
Allow	10.20.0.0/19	*	157.55.252.0/22	*	4
Allow	10.20.0.0/19	*	157.55.252.0/22	*	1
Deny	*	*	65.52.244.0/22	*	4

IP, Port, and Protocol: **bit vectors**

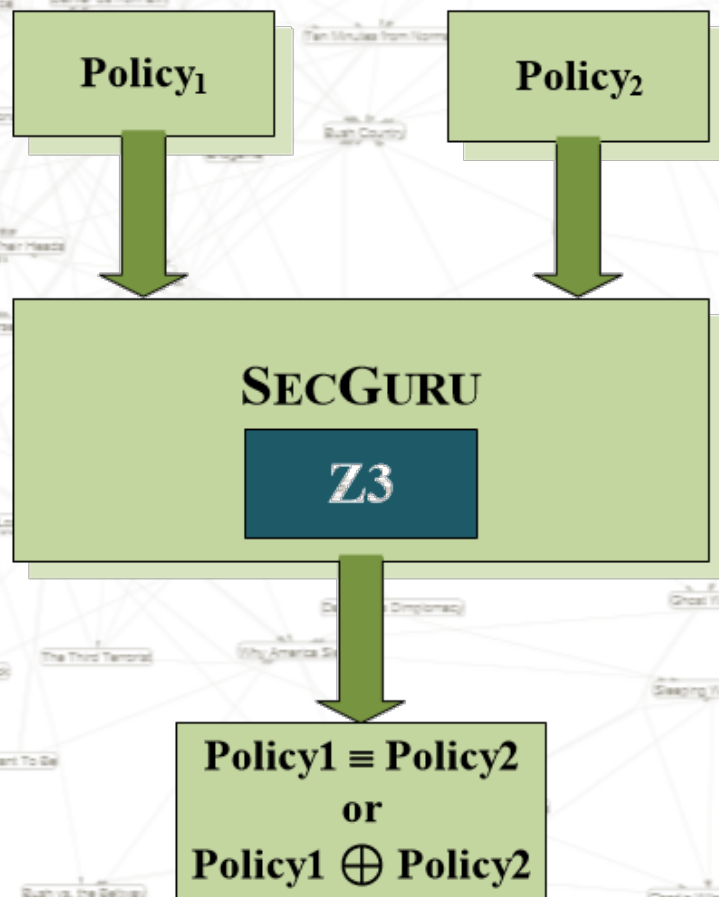
Policy: **Bit-vector logic**

Allow: $(10.20.0.0 \leq s$
 $rclp\ 10.20.31.255)$
 $\wedge (157.55.252.0 \leq dstIp$
 $< 157.55.252.255) \wedge$

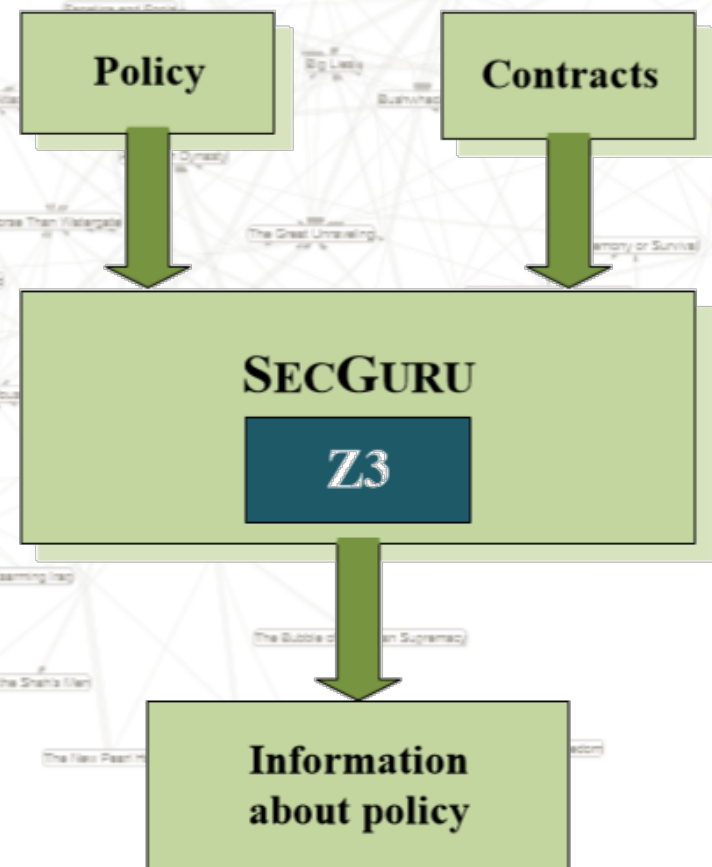
Policy $\downarrow 0 = false$
Policy $\downarrow i+1 = if$
IsAllow $\downarrow i$

Two uses of SecGuru

Change-Impact Analysis



Contract Validation



Two uses of SecGuru

Does policy permit outgoing traffic to **some** address in 65.52.244.0/22?

Query:

$(65.52.244.0 \leq dstIp \leq 65.52.247.255)$

Check Satisfiability of

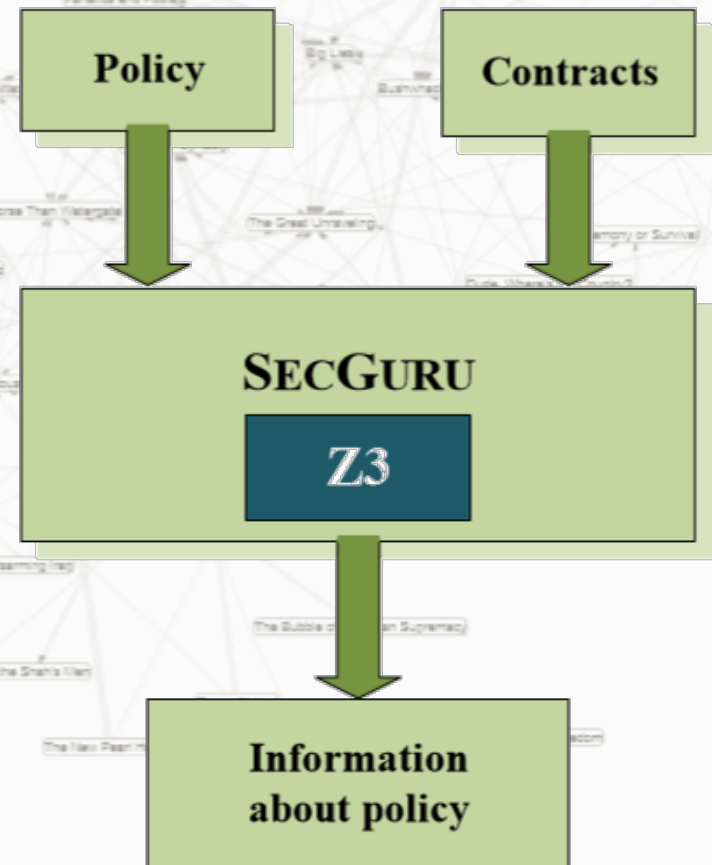
Query $\wedge$ **Policy**

Does policy permit connections to **all** the remote addresses in the range 65.52.244.0/22?

Check Unsatisfiability of

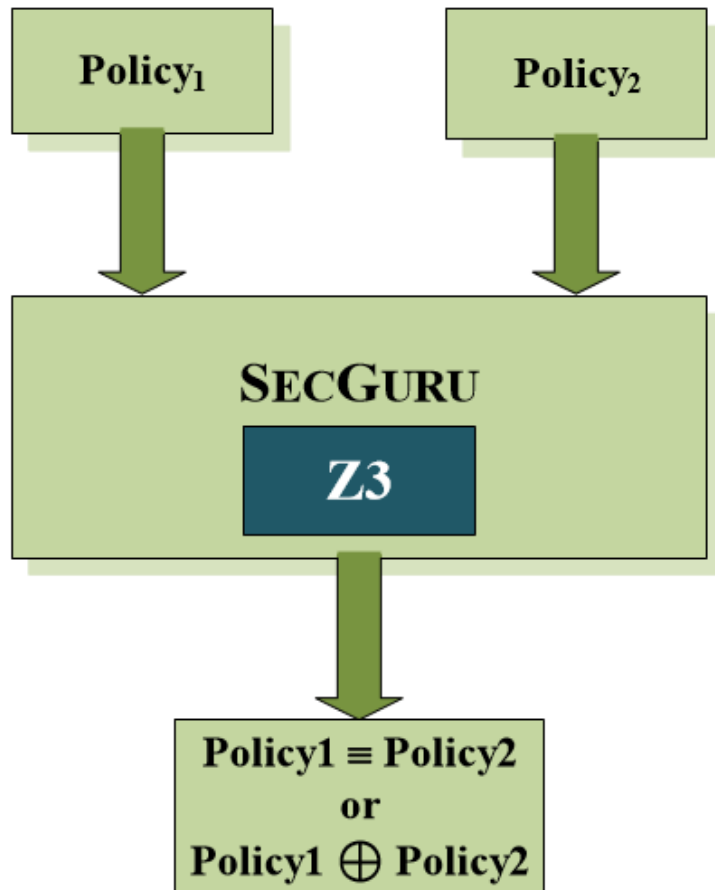
Query $\wedge \neg$ **Policy**

Contract Validation



Two uses of SecGuru

Change-Impact Analysis



Semantic diff between policies

Is ***Policy₁*** $\equiv$ ***Policy₂*** ?

If not, print ***Policy₁*** $\oplus$ ***Policy₂***

Traffic accepted by ***P₁***, but not ***P₂***.

Models for ***P₁*** $\wedge \neg$ ***P₂***

Traffic accepted by ***P₂***, but not ***P₁***.

Models for ***P₂*** $\wedge \neg$ ***P₁***

Two uses of SecGuru

Change-Impact Analysis

Semantic diff between policies

Policy₁



Policy₂

Policy₃

Traffic accepted by policy₁, but not policy₂

S.no	SrcIP	SrcPort	DstIP	DstPort	Protocol	TCP Flags	ICMP Type
1	10.25.252.0-10.25.252.255	*	239.0.0.0-239.255.255.255	*	IP		
2	10.62.12.0-10.62.12.255	*	168.63.161.0-168.63.161.15	80	TCP		
3	10.62.12.0-10.62.12.255	*	168.63.161.0-168.63.161.15	443	TCP		
4	10.62.12.0-10.62.12.255	*	168.63.161.0-168.63.161.15	8080	TCP		
5	10.146.192.0-10.146.192.255	*	239.0.0.0-239.255.255.255	*	IP		
6	210.126.9.11-210.126.9.12	*	168.63.161.0-168.63.161.15	80	TCP		
7	210.126.9.11-210.126.9.12	*	168.63.161.0-168.63.161.15	443	TCP		
8	210.126.9.11-210.126.9.12	*	168.63.161.0-168.63.161.15	8080	TCP		
9	216.52.28.197	*	168.63.161.0-168.63.161.15	80	TCP	SYN	
10	216.52.28.197	*	168.63.161.0-168.63.161.15	443	TCP	SYN	

Traffic accepted by policy₂, but not policy₁

S.no	SrcIP	SrcPort	DstIP	DstPort	Protocol	TCP Flags	ICMP Type
1	0.0.0.0-10.19.255.255	*	168.63.161.0-168.63.161.15	19999	TCP		
2	1.202.140.226-1.202.140.227	*	168.63.161.0-168.63.161.15	0-8549	TCP		
3	1.202.140.226-1.202.140.227	*	168.63.161.0-168.63.161.15	8552-8569	TCP		
4	1.202.140.226-1.202.140.227	*	168.63.161.0-168.63.161.15	8571-8588	TCP		
5	10.7.32.0-10.7.35.255	*	168.63.161.0-168.63.161.15	0-8549	TCP		
6	10.7.32.0-10.7.35.255	*	168.63.161.0-168.63.161.15	8552-8569	TCP		
7	10.7.32.0-10.7.35.255	*	168.63.161.0-168.63.161.15	8571-8588	TCP		
8	10.7.51.0-10.7.51.31	*	168.63.161.0-168.63.161.15	0-79	TCP		
9	10.7.51.0-10.7.51.31	*	168.63.161.0-168.63.161.15	81-442	TCP		
10	10.7.51.0-10.7.51.31	*	168.63.161.0-168.63.161.15	444-8079	TCP		

All-BVSAT: A compact model enumeration

Really naïve model enumeration:

- To generate the $(k+1)st$ model, negate all the k models seen so far

- $(\bigvee i \uparrow Allow \downarrow i) \wedge (\bigwedge j \uparrow \neg Den$
 $y \downarrow j) \wedge (\bigwedge k \uparrow \neg Model \downarrow k) \quad \dots$

$2^{32+16+32+16}$ models

Smarter model enumeration in SecGuru using All-BVSAT (idea):

- Find initial $srcIp \downarrow 0, srcPort \downarrow 0 \models \mathbf{Policy} \downarrow 1 \wedge \neg \mathbf{Policy} \downarrow 2$

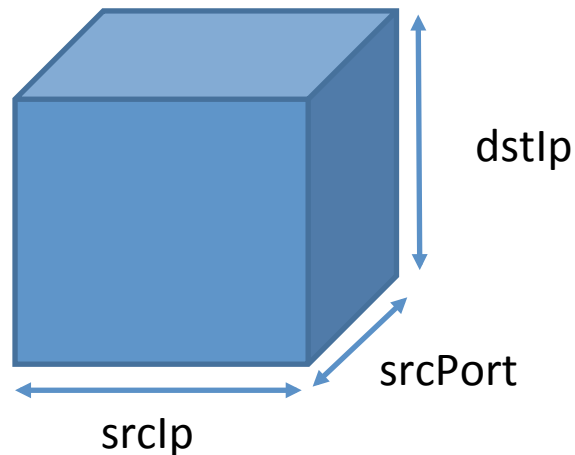
Maximize bounds $lo/srcIp \leq srcIp \leq hi/srcIp$.

All-BVSAT: A compact model enumeration

Maximize bounds:

$$\begin{aligned} & \blacksquare \blacksquare lo \downarrow srcIp \leq srcIp \leq hi \downarrow srcIp \wedge l \\ & o \downarrow srcPort \leq srcPort \leq hi \downarrow srcPort \wedge lo \downarrow dstIp \\ & \leq dstIp \leq hi \downarrow dstIp \ \&F \& \mathbf{Policy} \downarrow \mathbf{1} \wedge \neg \mathbf{Polic} \\ & \mathbf{y} \downarrow \mathbf{2} \end{aligned}$$

Result is a *cube*:

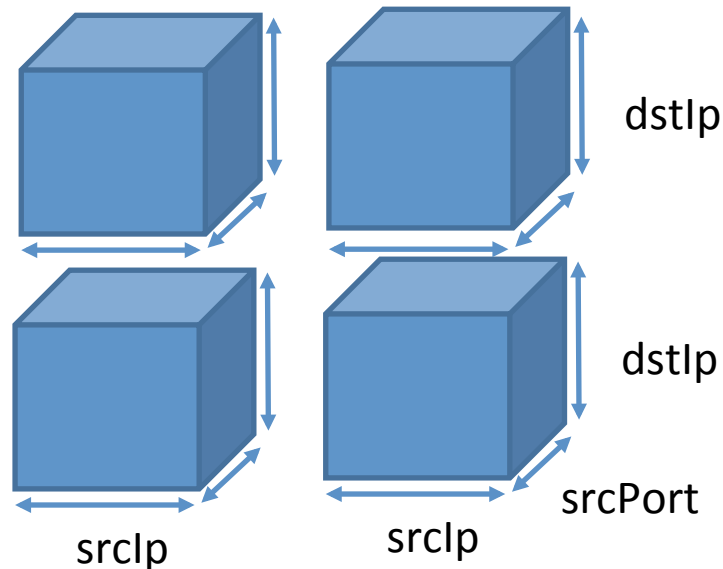


All-BVSAT: A compact model enumeration

More succinct: Maximize *multiple* bounds

$$\blacksquare \blacksquare (lo_1 \leq srcIp \leq hi_1 \vee lo_2 \leq srcIp \leq hi_2) \wedge lo_1 \leq srcPort \leq hi_1 \wedge lo_2 \leq srcPort \leq hi_2 \wedge (lo_3 \leq dstIp \leq hi_3 \vee lo_4 \leq dstIp \leq hi_4) \wedge \text{Policy}_1 \wedge \neg \text{Policy}_2$$

Result is a *multi-cube*:





Long ago in a network far away the rebel Hassel forces began to claim

Do old-school tools apply to Network Verification?

- Experiments by Nuno Lopes, March-May 2013
- Queries over Stanford Network + Extensions
- Observation: Datalog is OK to express network queries.
- Query tools tried out.
 - Datalog + Hassel algebra
 - PDR engine
 - Hardware model checkers (ABC, IIMC, IC3)
 - Bounded model checking. Two encodings.
 - Symbolic simulation

Very preliminary finding: PDR ~ Hassel, others work on small (Stanford) network. Hardware tools choked.

Summary

An Introduction to SMT with Z3

Algorithmic principles of SAT/SMT

Theories, Solvers and Applications