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/

Munich, Germany -גר ב

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

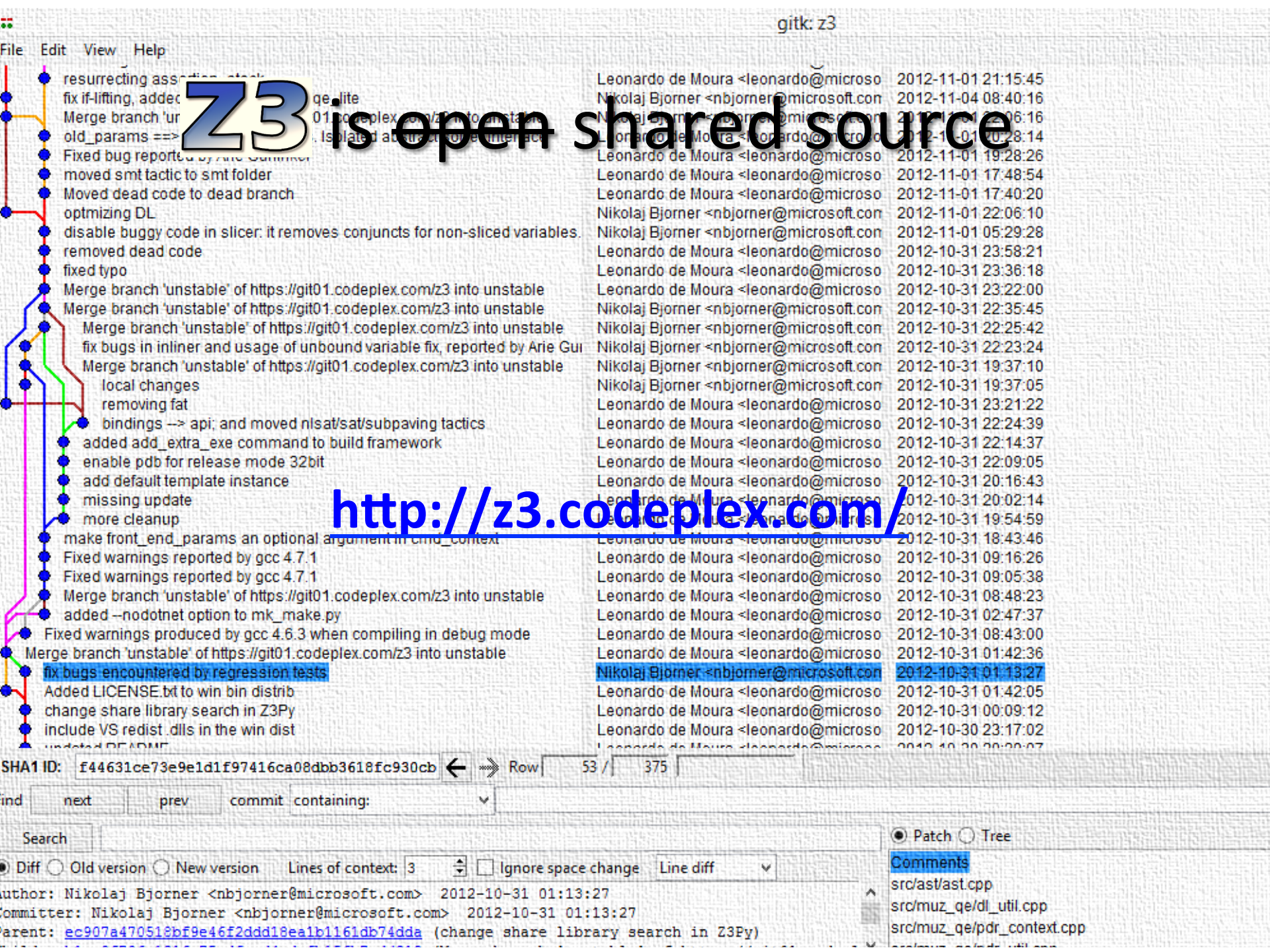
"Z3 is now open source. The source code is available under the MSR-LA license. This is the same license used to distribute the Z3 binaries. The source code can be used for non-commercial purposes. "

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.

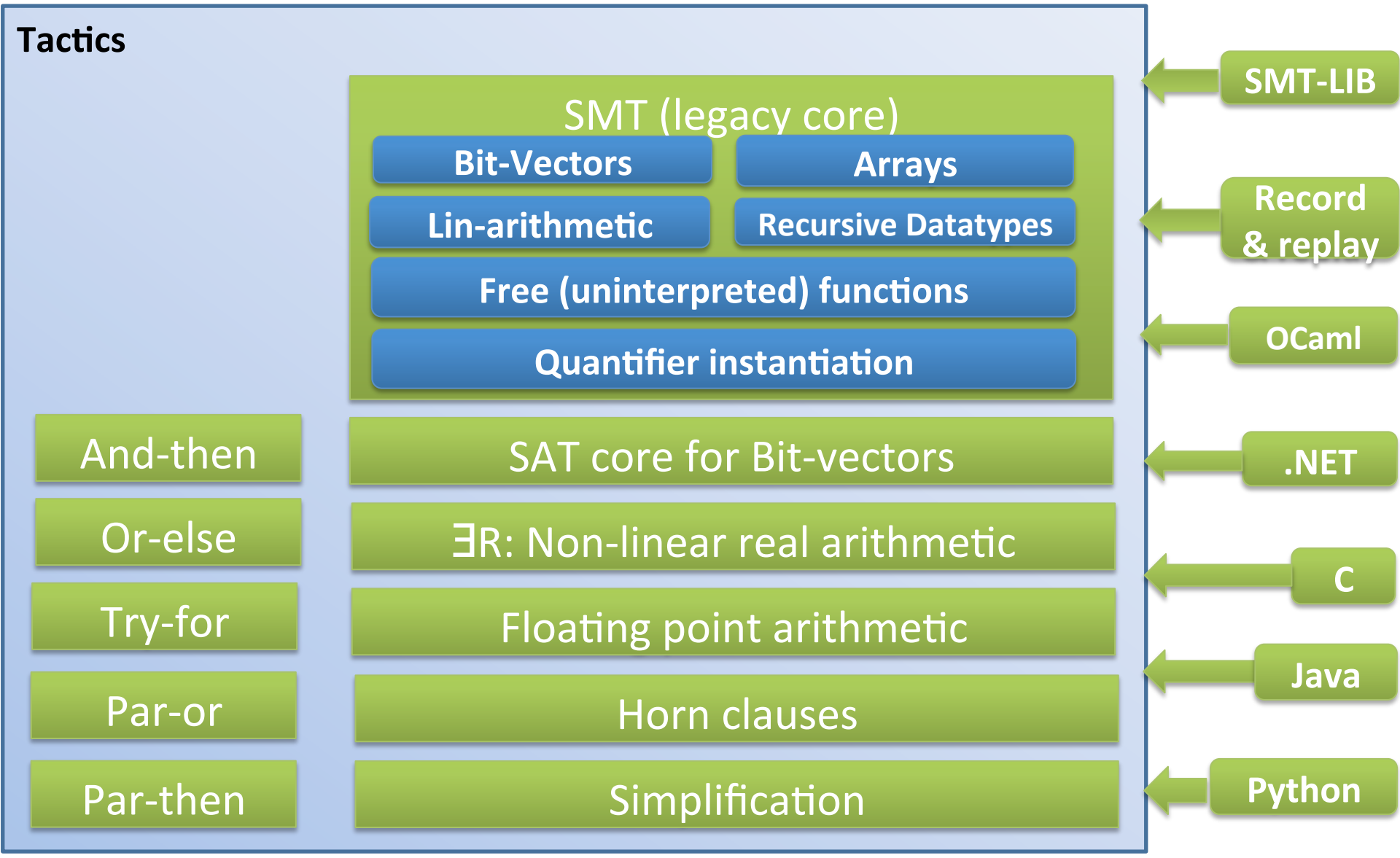
Open Source means Open Source. And that means no limitations on use and reuse.

« Open Sourcing Z3

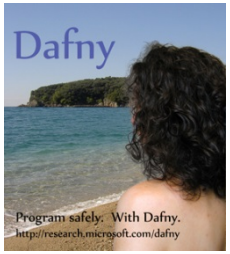
Leonardo de Moura' Homepage


gitk: z3
File Edit View Help
Z3 is open shared source
http://z3.codeplex.com/
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
Patch Tree
Comments
src/ast/ast.cpp
src/muz_qe/dl_util.cpp
src/muz_qe/pdr_context.cpp
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)

Z3 architecture - new



Some Microsoft Tools based on Z3



Program Verification

Vcc

HAVOC

Auditing

BOOGIE

Type Safety

SLAyer

SLAM

Over-Approximation

TERMINATOR

Yogi

Under-Approximation

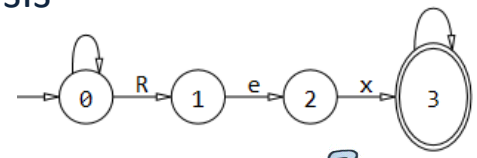
SAGE

Pex

Testing

M3

Analysis



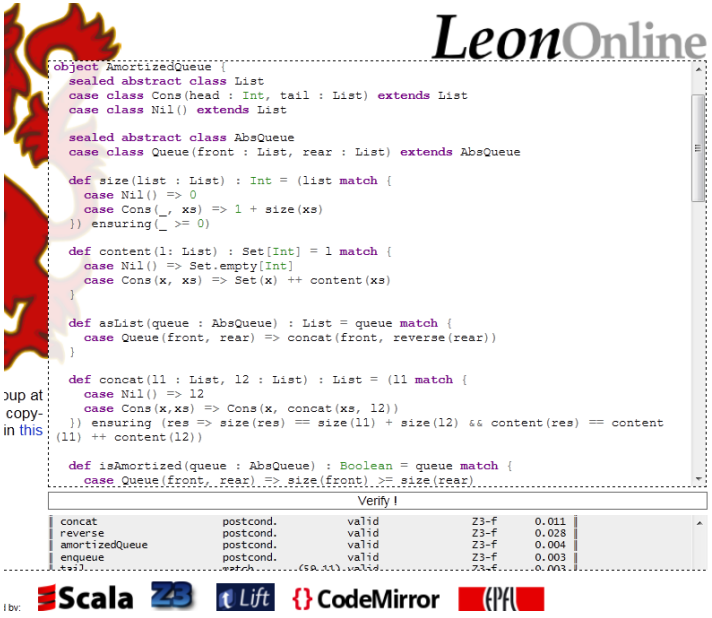
Synthesis

FORMULA
Modeling Foundations.

Cool tools using Z3

Sledge Hammer

Liquid Types



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

Old introduction slide to SAGE & Pex

State today:

Pex is mature & Moles in VS2012

SAGE actively used internally at Microsoft

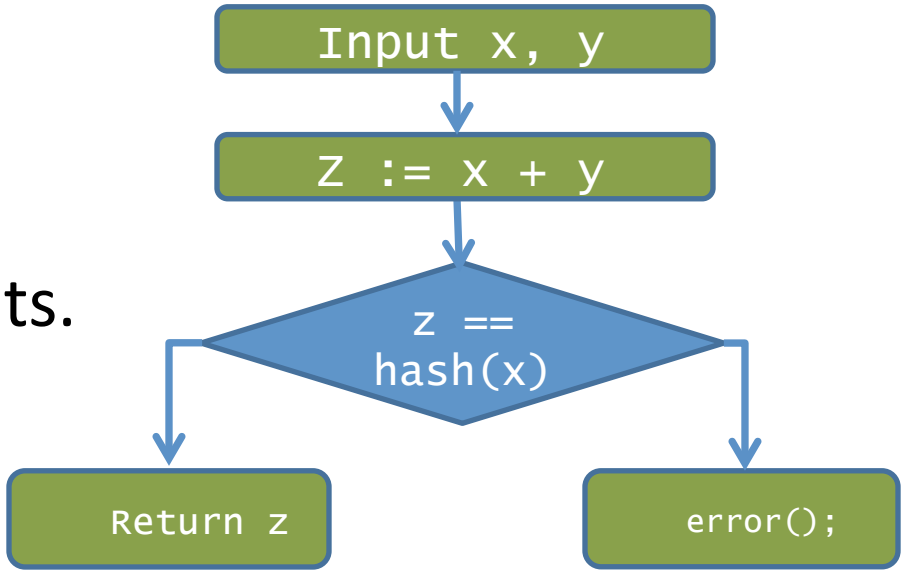
Method: Dynamic Test Generation

Run program with **random** inputs.

Gather constraints on inputs.

Use constraint solver to generate new inputs.

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



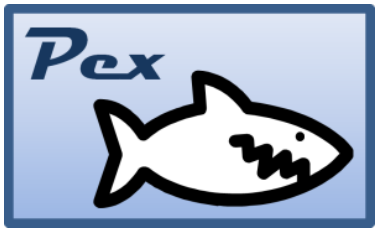
Can't statically generate value for x that satisfy "z==hash(x)"

But we can solve for y in:

x + y = hash(x)

Fuzzing and Test Case Generation

SAGE



Internal. For Security Fuzzing

Runs on x86 instructions

External. For Developers

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 Securi

Runs on x86 instru

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** …"*

m

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

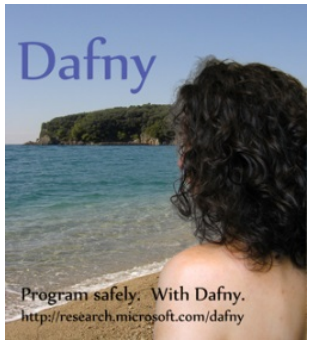
Millions of **$** saved for Users and Microsoft

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

3+ Billion constraints **-** largest usage for any SMT solver

Adapted from [Patrice Godefroid, ISSTA 2010]

Verification
It is cool
easy
fun
attractive



Program Verification

Vcc

HAVOC

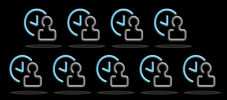
Auditing

BOOGIE

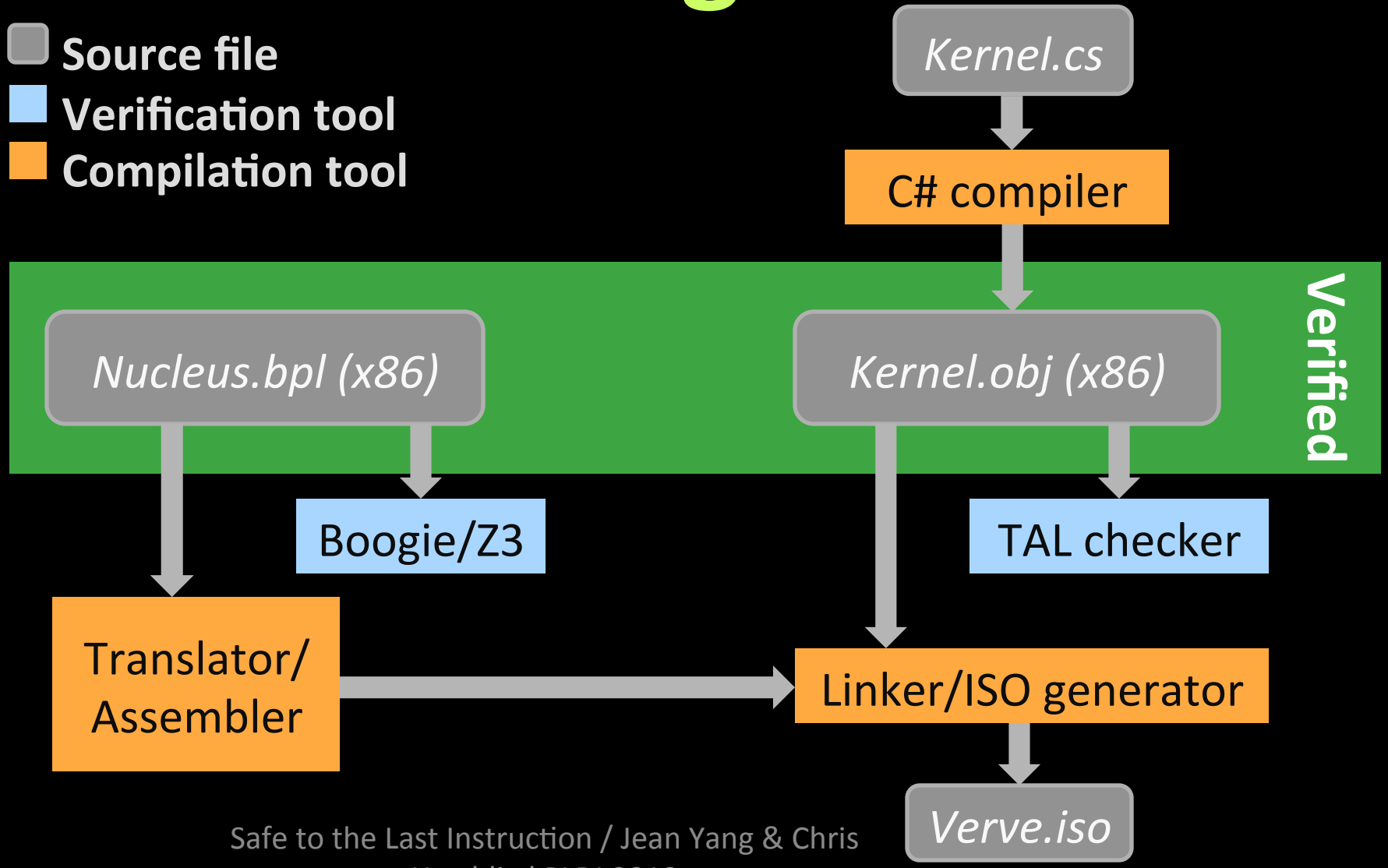
Type Safety

SLAyer

Building Verve



9 person-months



Safe to the Last Instruction / Jean Yang & Chris Hawbliztl PLDI 2010

Validation

SecGuru: Automatic Validation of Network Connectivity Restrictions

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

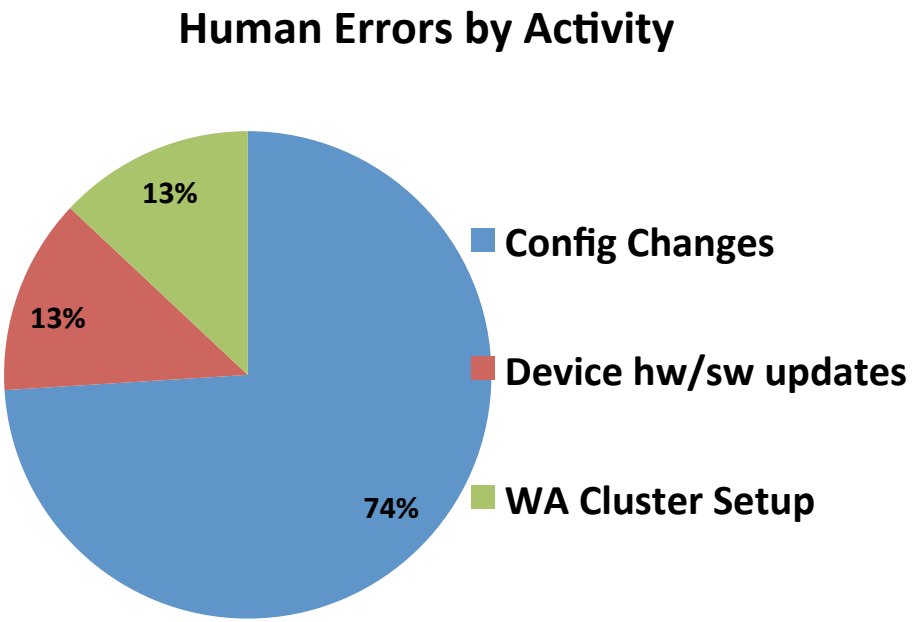
- Do devices enforce policy?
- Ripple effect of policy changes

Arcane

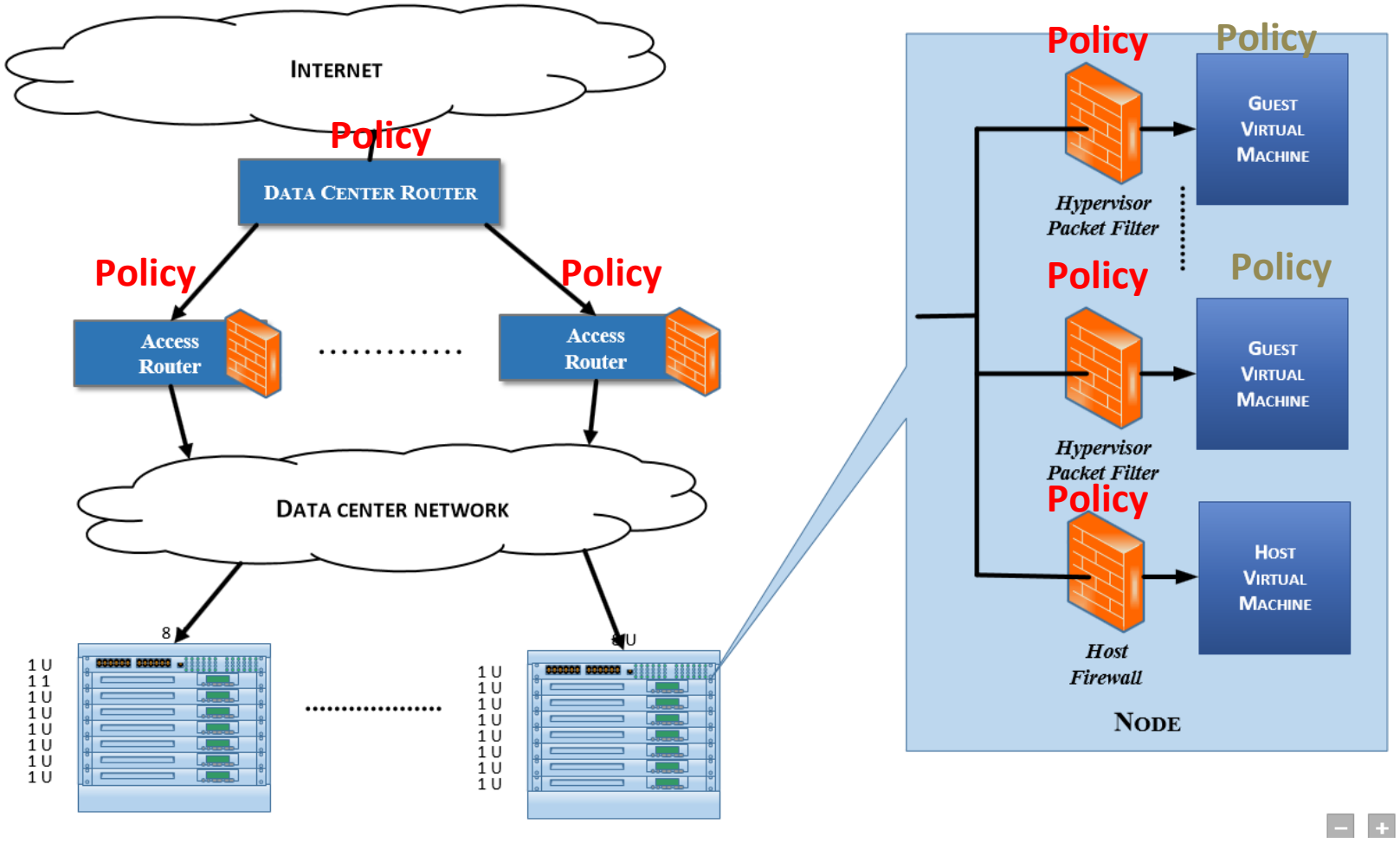
- Low-level configuration files
- Mostly manual effort
- Kept working by

"Masters of Complexity"

Human errors > 4 x DOS attacks



A Data-center Architecture

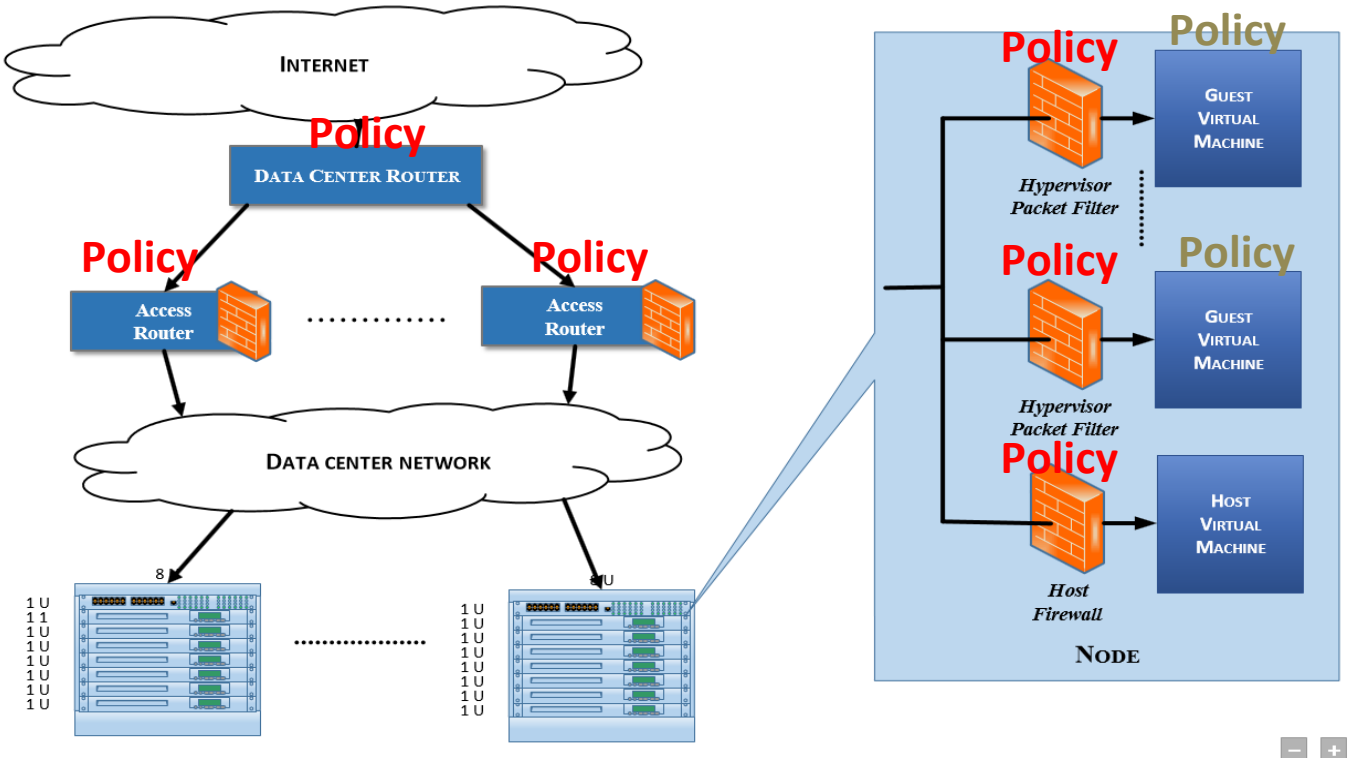


A Data-center Architecture

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

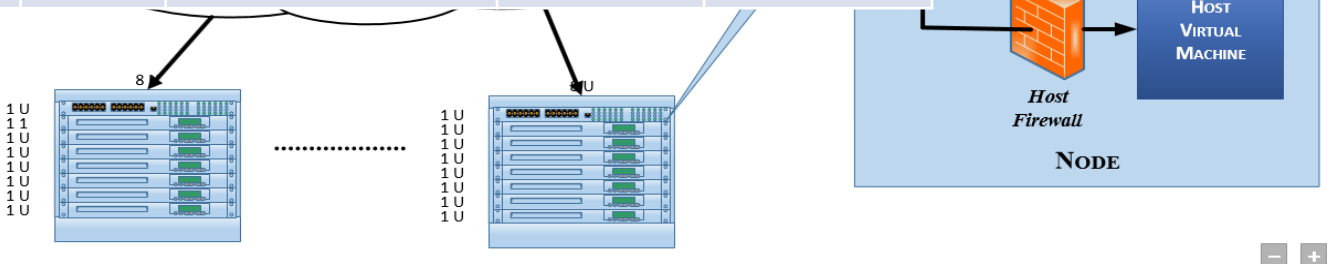


Network Policies
What they look like in routers

Defense in Depth = Crypto + Policies **- outside IP can be spoofed**

Efficient and Flexible Defense by Policy only **- inside IP cannot be spoofed**

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



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

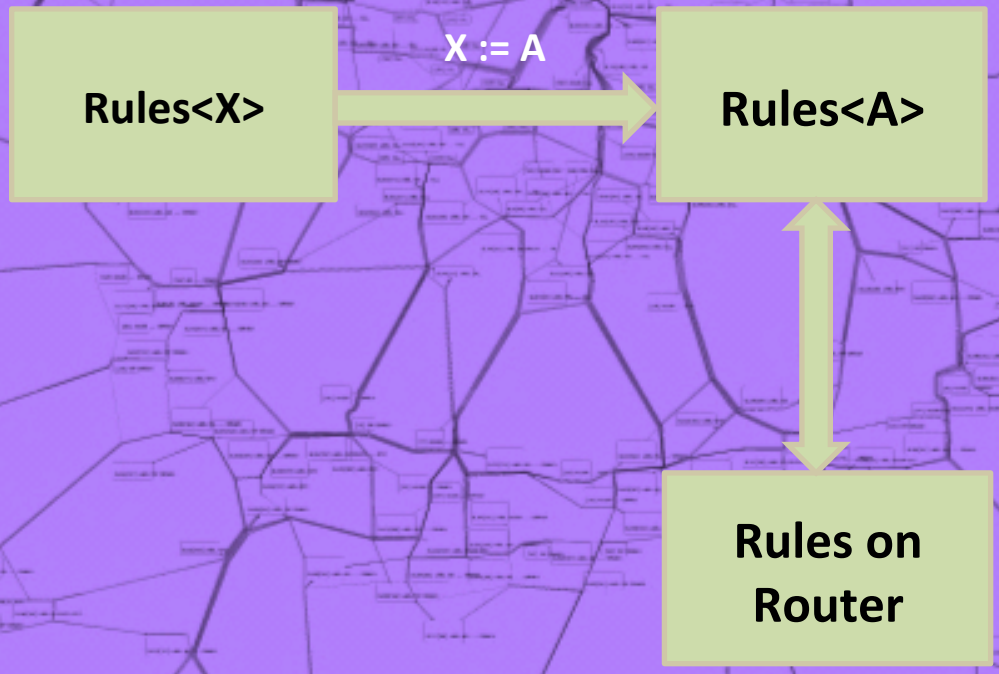
Hardware – capacity, semantics

New Services – *check effect of new rules*

Patches – *check for regressions*

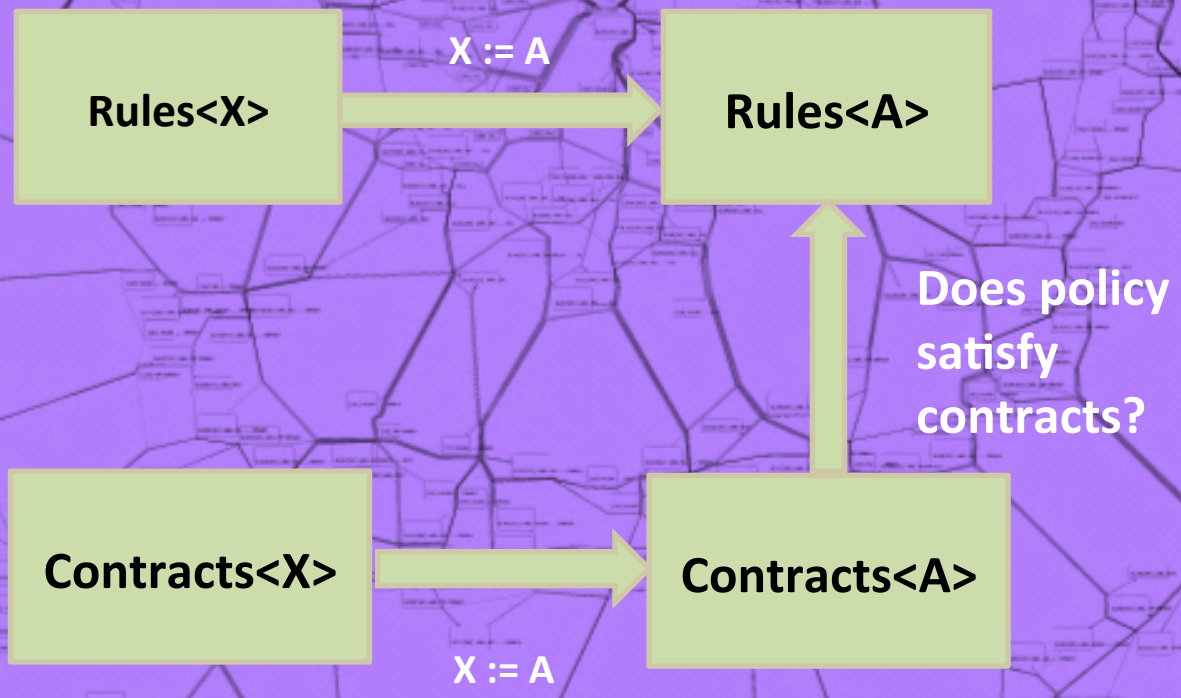
Towards automation:
Checking Policy Configuration with **SecGuru**

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

$$\blacksquare Allow:\& \blacksquare (10.20.0.0 \leq srcIp\ 10.20.31.255) \wedge (157.55.252.0 \leq dstIp \leq 157.55.252.255) \wedge$$

$$Policy : (Vi \uparrow \blacksquare Allow \downarrow i) \wedge (\wedge j \uparrow \blacksquare \neg Den$$

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

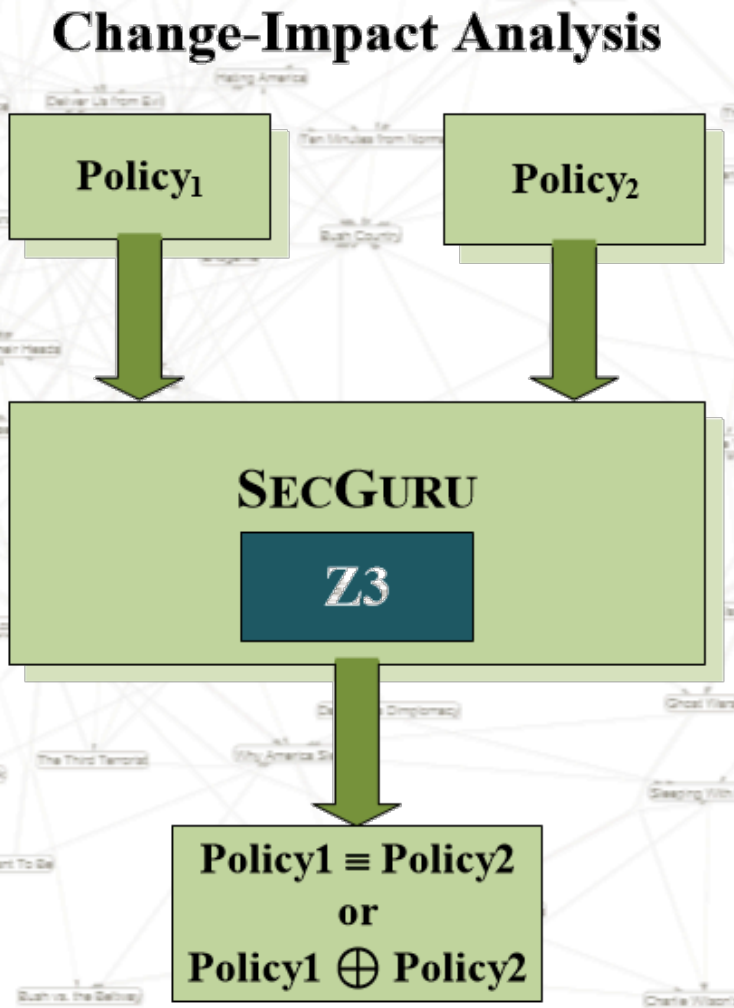
IP, Port, and Protocol: bit vectors
Policy: Bit-vector logic

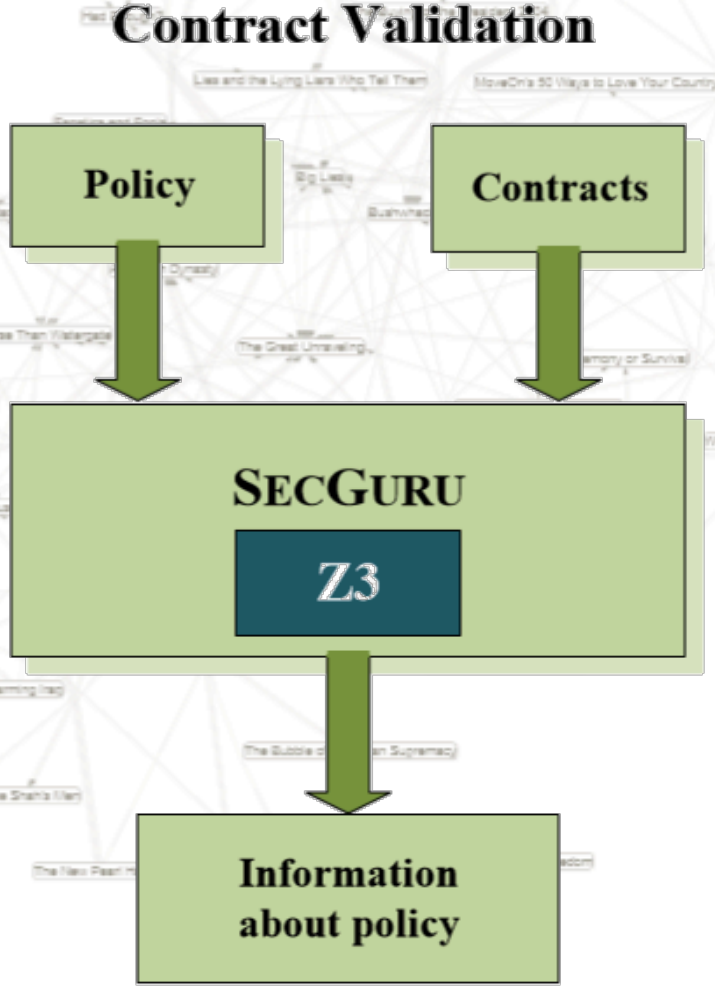
$$\blacksquare Allow{:}\& \blacksquare(10.20.0.0 \leq srcIp\ 10.20.31.255) \wedge (157.55.252.0 \leq dstIp \leq 157.55.252.255) \wedge$$

$$Policy\downarrow 0 = false$$

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

Two uses of **SecGuru**





Two uses of **SecGuru**

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

Query:
*(*65.52.244.0≤*dstIp*≤65.52.247.255*)*

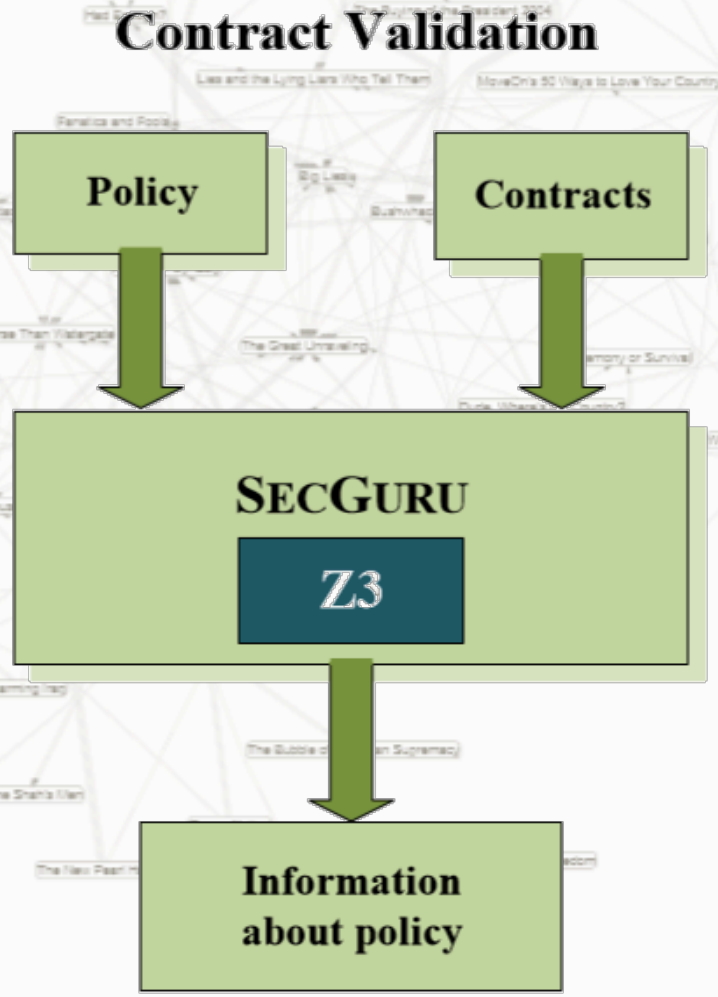
Check Satisfiability of

Query*∧*Policy

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

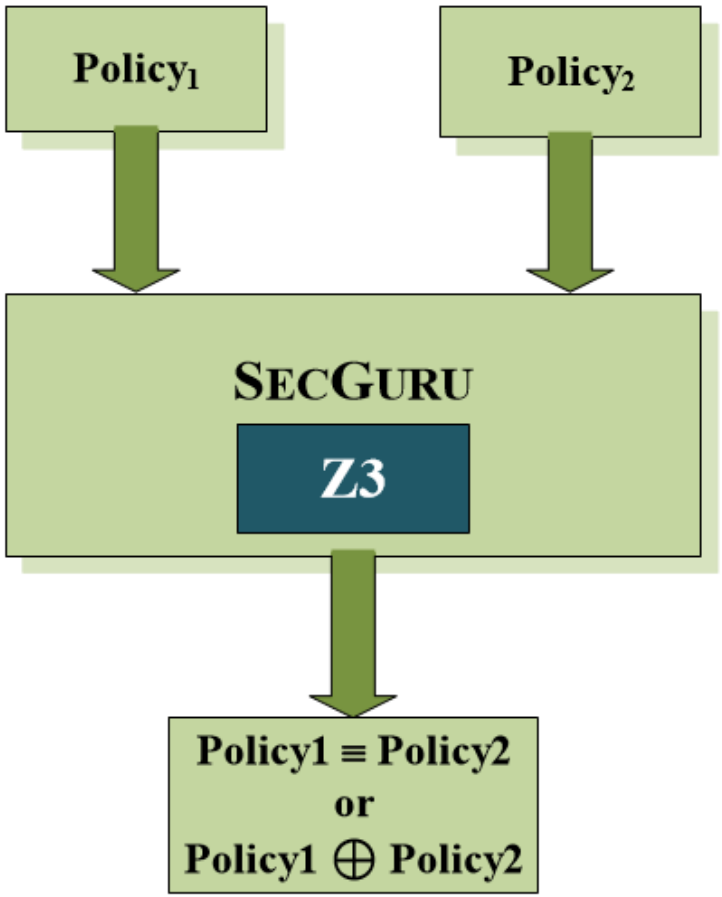
Check Unsatisfiability of

Query*∧¬*Policy



Two uses of **SecGuru**

Change-Impact Analysis



Semantic diff between policies

Is $Policy{\downarrow}1 \equiv Policy{\downarrow}2$?

If not, print $Policy{\downarrow}1 \oplus Policy{\downarrow}2$

Traffic accepted by $P{\downarrow}1$, but not $P{\downarrow}2$.

Models for $P{\downarrow}1 \wedge \neg P{\downarrow}2$

Traffic accepted by $P{\downarrow}2$, but not $P{\downarrow}1$.

Models for $P{\downarrow}2 \wedge \neg P{\downarrow}1$

Two uses of **SecGuru**

Change-Impact Analysis

Semantic diff between policies


```
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;, TC
8, 210.126.9.11-210.126.9.12;, *;, 168.63.161.0-168.63.161.15;, 8080;, T
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
3, 1.202.140.226-1.202.140.227;, *;, 168.63.161.0-168.63.161.15;, 8552-8
4, 1.202.140.226-1.202.140.227;, *;, 168.63.161.0-168.63.161.15;, 8571-6
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;,
7, 10.7.32.0-10.7.35.255;, *;, 168.63.161.0-168.63.161.15;, 8571-65535;,
8, 10.7.51.0-10.7.51.31;, *;, 168.63.161.0-168.63.161.15;, 0-79;, TCP, S
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;, T
```

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
- $(Vi\uparrow\blacksquare Allow\downarrow i)\wedge(\wedge j\uparrow\blacksquare\neg Den y\downarrow j)\wedge(\wedge k\uparrow\blacksquare\neg Model\downarrow k)$ …
  $2\uparrow 32+16+32+16$ models

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

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

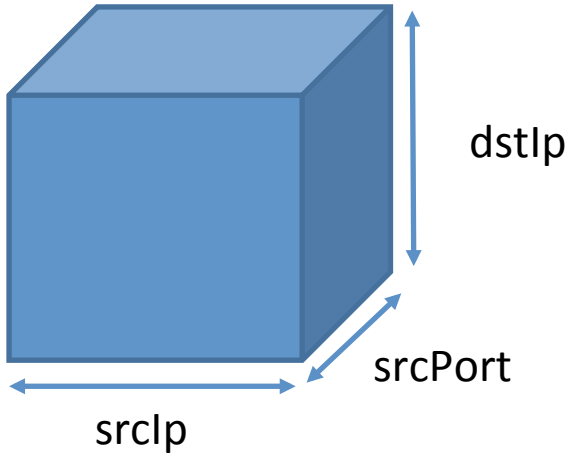
- Maximize bounds $lo \mid srcIp \leq srcIp \leq hi \mid srcIp$;

All-BVSAT: A compact model enumeration

Maximize bounds:

$$\blacksquare\blacksquare lo{\downarrow}srcIp \leq srcIp \leq hi{\downarrow}srcIp \wedge lo{\downarrow}srcPort \leq srcPort \leq hi{\downarrow}srcPort \wedge lo{\downarrow}dstIp \leq dstIp \leq hi{\downarrow}dstIp \;\&\models\& \boldsymbol{Policy{\downarrow}1} \wedge \neg \boldsymbol{Policy{\downarrow}2}$$

Result is a *cube:*

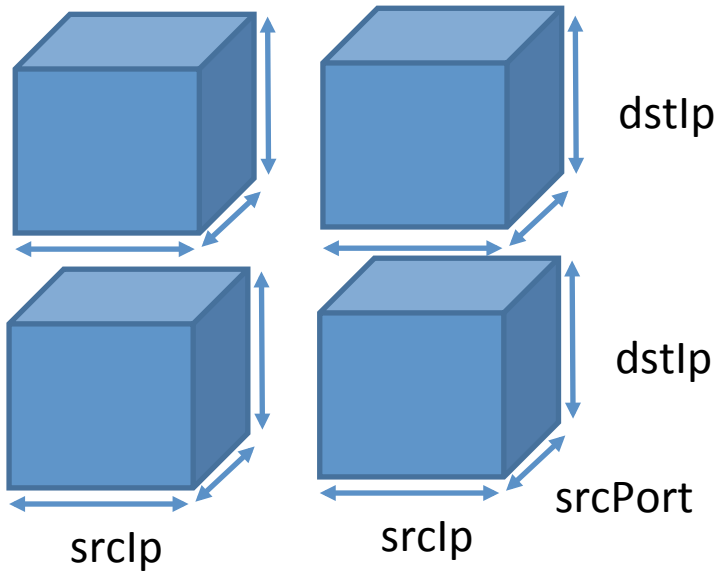


All-BVSAT: A compact model enumeration

More succinct: Maximize *multiple* bounds

$$\blacksquare\blacksquare(lo{\downarrow}1 \le srcIp \le hi{\downarrow}2 \vee lo{\downarrow}2 \le srcIp \le hi{\downarrow}2) \wedge lo{\downarrow}srcPort \le srcPort \le hi{\downarrow}srcPort \wedge (lo{\downarrow}3 \le dstIp \le hi{\downarrow}3 \vee lo{\downarrow}4 \le dstIp \le hi{\downarrow}4) \;\&\models\& \boldsymbol{Policy{\downarrow}1} \wedge \neg \boldsymbol{Policy{\downarrow}2}$$

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 chocked.

Summary

An Introduction to SMT with Z3

Algorithmic principles of SAT/SMT

Theories, Solvers and Applications