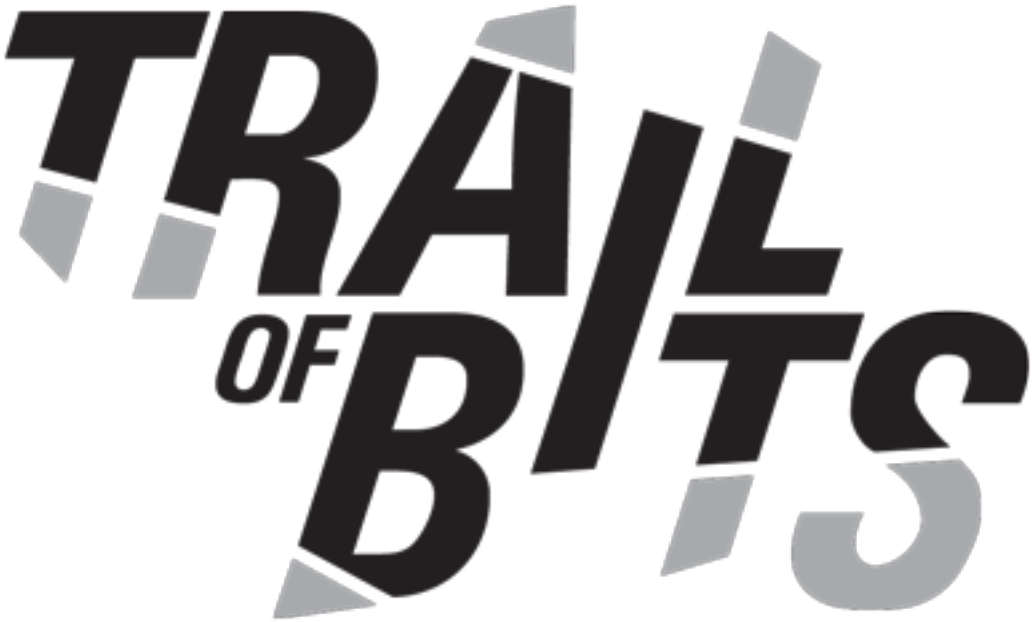
Making a Scalable Automated Hacking System

From DevOps to Pwning

Artem Dinaburg
artem@trailofbits.com

About Me



Talk Themes

- ToB in the Cyber Grand Challenge
- New approaches to bug finding
- Discussions about distributed systems and automated patching

In the beginning…

- Andrew: “Hey, you want to lead our CGC team?”

In the beginning…

- Andrew: “Hey, you want to lead our CGC team?”
- Me: “I don’t know…”

In the beginning…

- Andrew: “Hey, you want to lead our CGC team?”
- Me: “I don’t know…”
- Andrew: “Do ittt”

In the beginning…

- Andrew: “Hey, you want to lead our CGC team?”
- Me: “I don’t know…”
- Andrew: “Do ittt”
- Me: “Ok, lets do this!”

What is CGC?

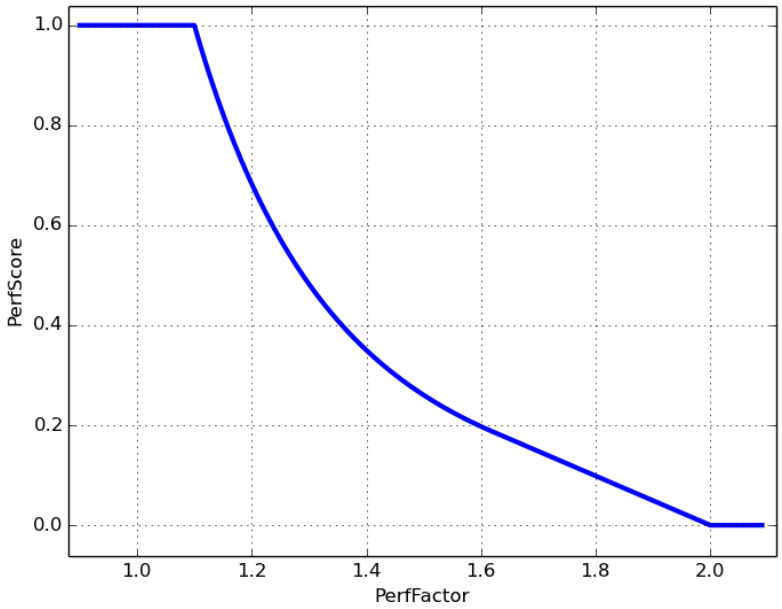
- It's a competition!
- To automate bug finding and patching
- In binary-only software
- Simplified OS: DECREE

Scoring

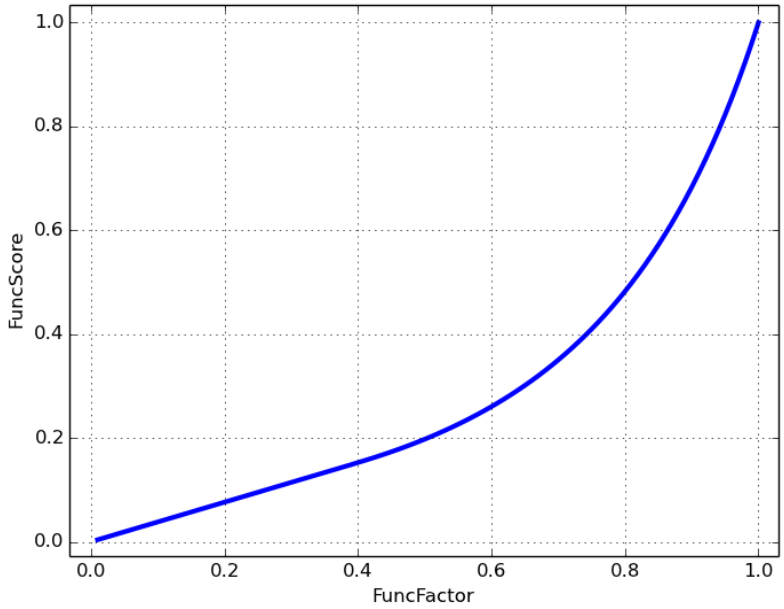
- Gain points for patching bugs.
  - –You **must** patch to score.
- Gain points for finding bugs.
  - –Only counts if you patch

Scoring

- Score is scaled by performance
  - Disk, Memory, CPU usage



(a) PerfFactor to PerfScore conversion curve.



(b) FuncFactor to FuncScore conversion curve.

The Competition

FOR ALL SECURE

BLOG

David Brumley

CEO, CO-FOUNDER

Thanassis Avgerinos

CO-FOUNDER

Alex Rebert

CO-FOUNDER

The Competition

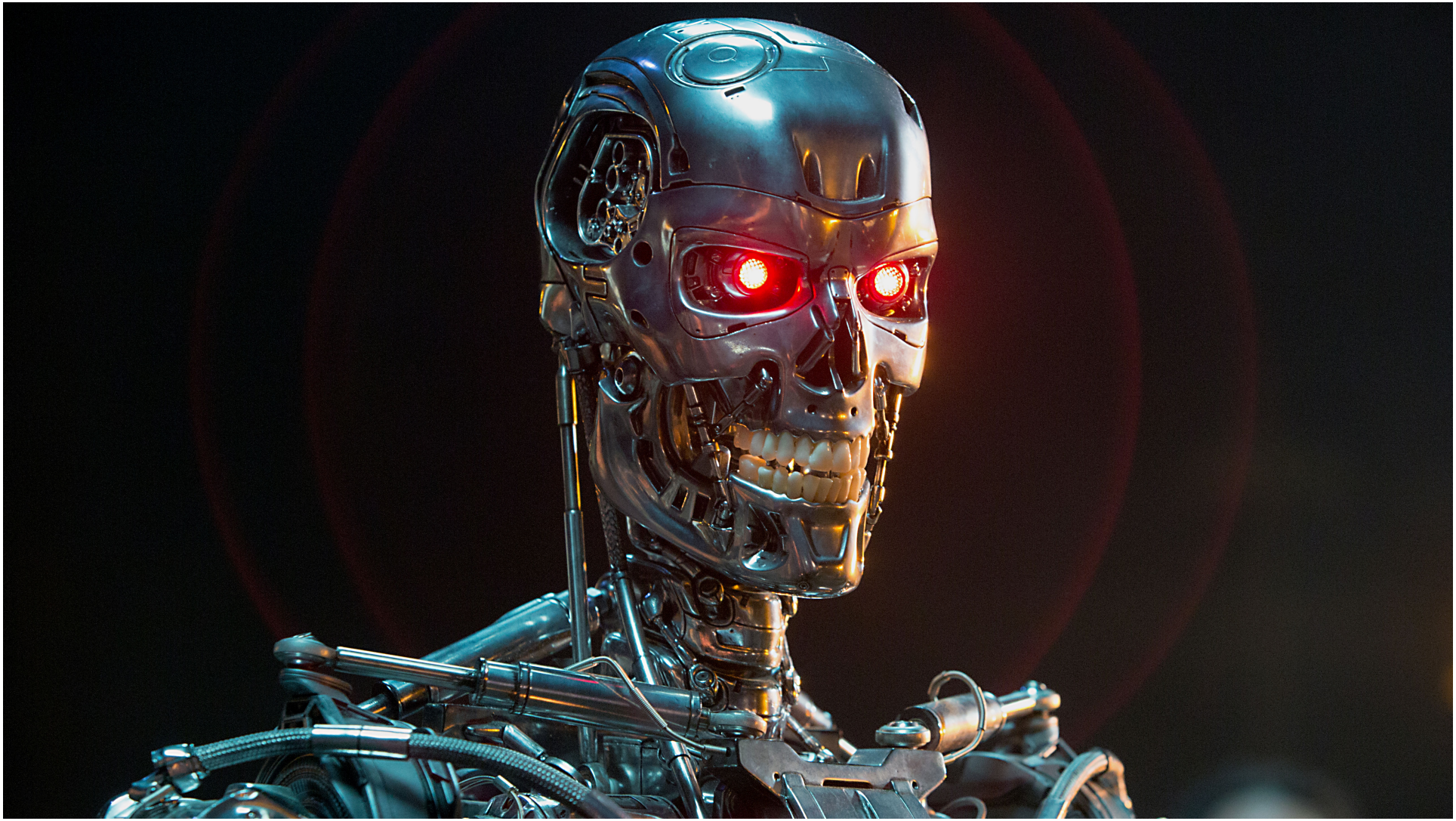
2012 IEEE Symposium on Security and Privacy

Unleashing MAYHEM on Binary Code

Sang Kil Cha, Thanassis Avgerinos, Alexandre Rebert and David Brumley
Carnegie Mellon University
Pittsburgh, PA
{*sangkilc, thanassis, alexandre.rebert, dbrumley*}*@cmu.edu*

Goal: Winning

Our Solution: Cyberdyne



Step 1: Get Help

Peter

Felipe

Artem

Car

Step 2: Wizards

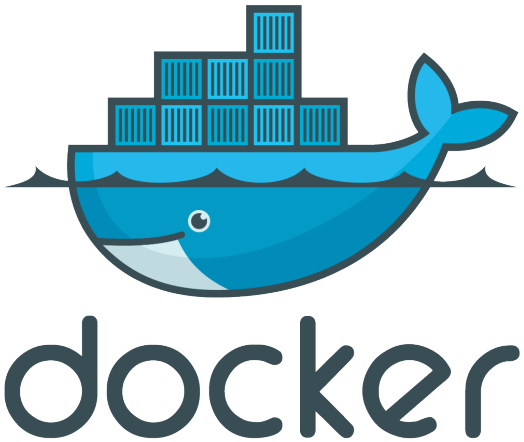
What to Build

First Quest: DevOps

Why DevOps?

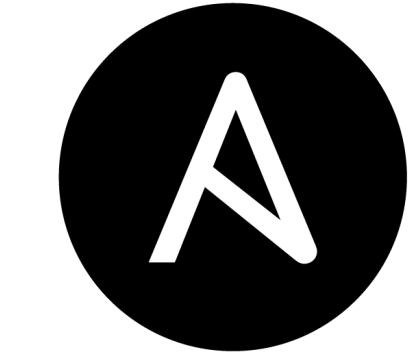
- Automate Everything
  - Hard Time Limit
  - Few People Resources
- Repeatable
  - “Works on my machine!”
- Faster Test Cycle
  - Focus on building tools, not doing IT

DevOps: Technology Stack



DevOps: Technology Stack

DevOps: Technology Stack



DevOps: Technology Stack

DevOps: Progress

The First Scored Event

- Scored Event 1. December 2014
- 6 Months to go.
- ToB Score: …

The First Test

- Scored Event 1. December 2014
- 6 Months to go.
- ToB Score:

0

Ask The Wizards, Again

- Me: Wizards, we are in trouble

Ask The Wizards, Again

- Me: Wizards, we are in trouble
- Wizards: Have you tried scoring points?

Lets Find Bugs!

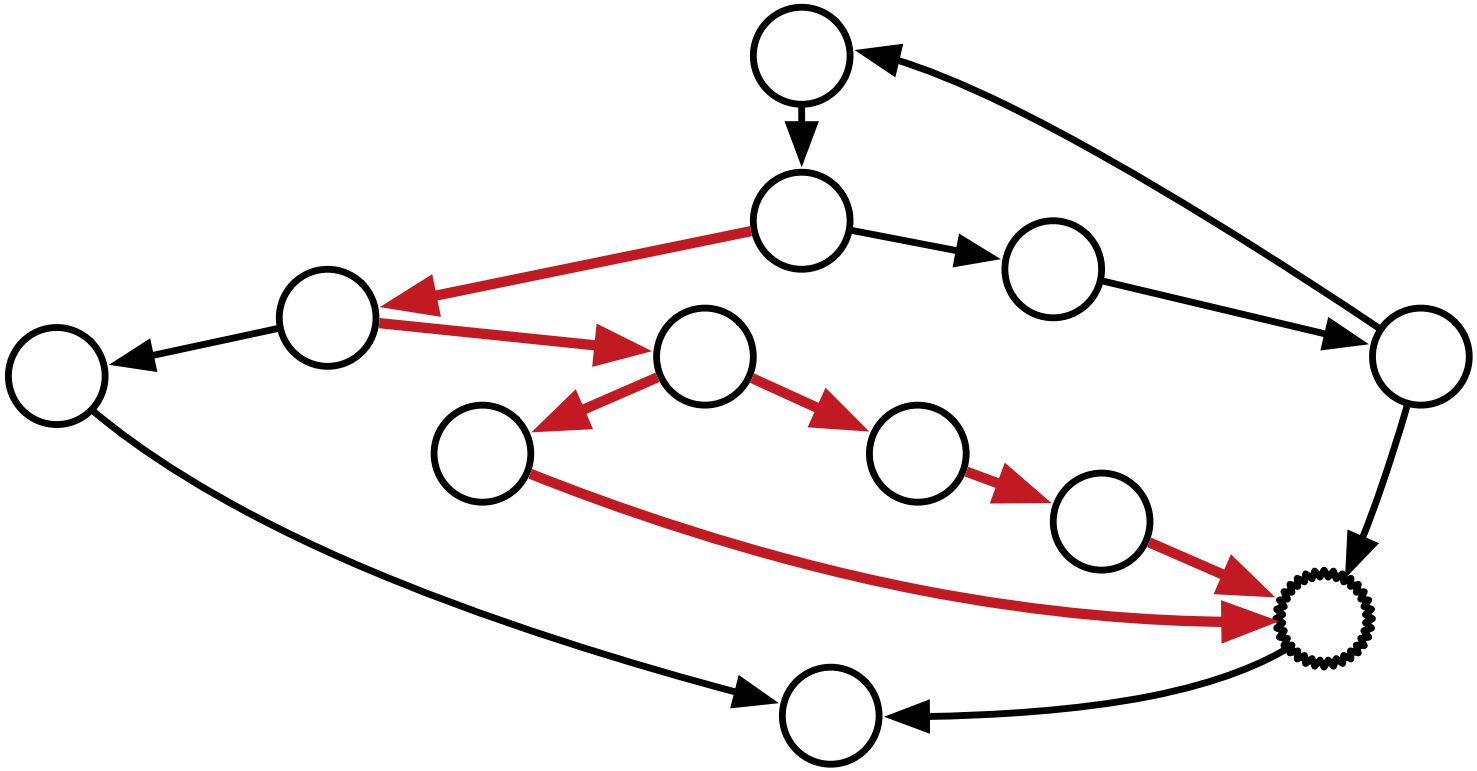
What's a Bug?

- Memory Safety Violations
  - Out of Bounds Read
  - Out of Bound Write
  - Use-After-Free
- Not Bugs (for CGC CQE)
  - Authentication bypass
  - Race Conditions
  - Permissions Problems
  - FPU Exceptions

We Needed Some Magic

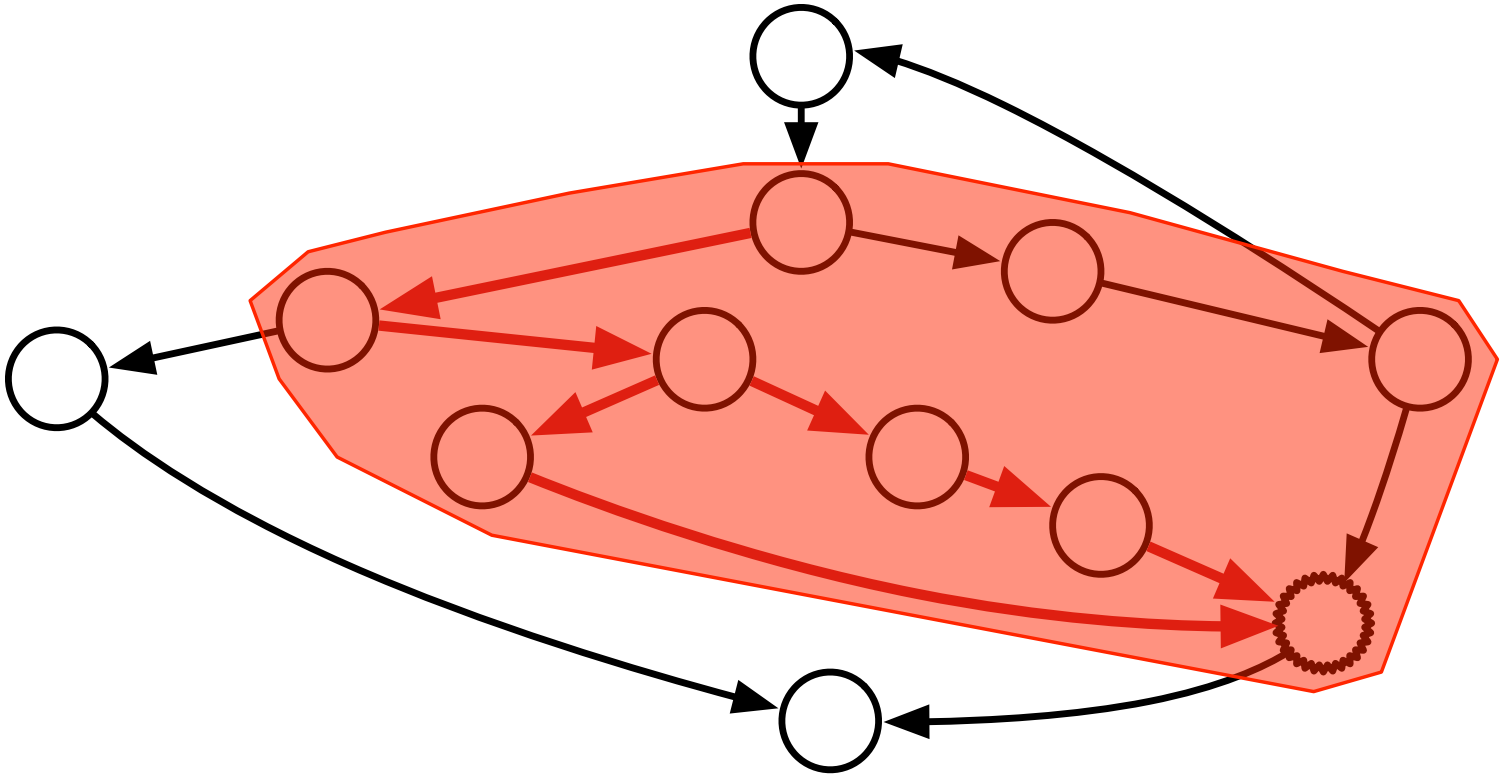
Bug Finding Theory

- No analysis will find all the bugs.
- Provably impossible.



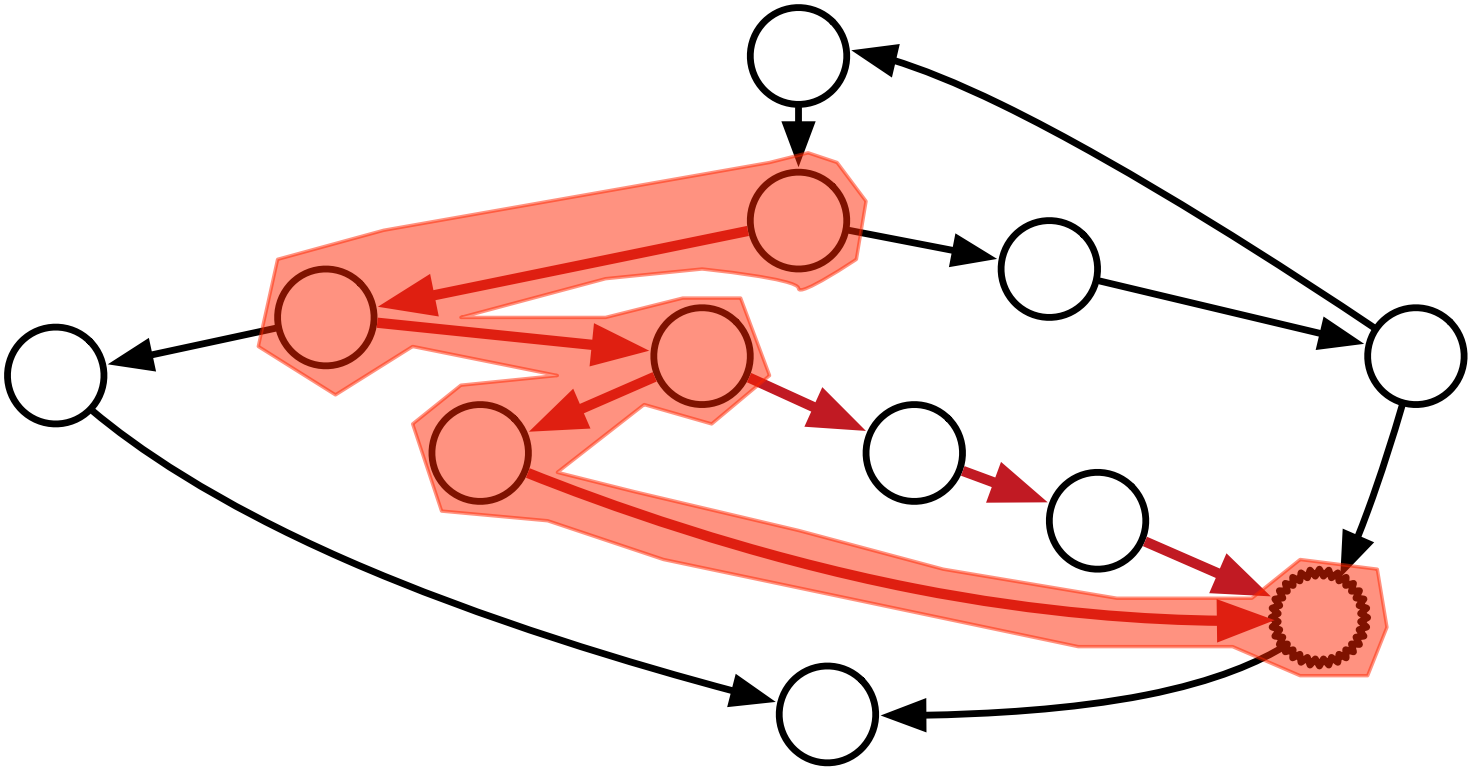
Bug Finding Theory

- Over Approximate Analyses
  - Points To, Abstract Interpretation



Bug Finding Theory

- Under Approximate Analyses
  - Fuzzing, Symbolic Execution



Under-Approximate Analyses: Roadblocks

- Different Roadblocks:
  Hard For Fuzzing, Easy for Symbolic Execution

```
if(input[0] == 0xBADFOOD)
```

Under-Approximate Analyses: Roadblocks

- Different Roadblocks:
  Hard for Symbolic Execution, Easy for Fuzzing

```
if(hash(input[0])
      == hash(input[1]))
```

Under-Approximate Analyses: Theory

- All tools operate over the **same domain**
- All discoveries are equally true
- What if we could share discoveries?

Analysis Boosting

- Sharing discoveries across tools creates a virtuous cycle that **removes roadblocks**

```
if(input[0] == 0xBADFOOD)
  if(hash(input[0])
    == hash(input[1]))
      BUG();
```

Analysis Boosting

- Sharing discoveries across tools creates a virtuous cycle that **removes roadblocks**

~~`if(input[0] == 0xBADF00D)`~~

```
if(hash(input[0])
   == hash(input[1]))
     BUG();
```

Analysis Boosting

- Sharing discoveries across tools creates a virtuous cycle that **removes roadblocks**

```
if(input[0] == 0xBADF00D)
  if(hash(input[0])
    == hash(input[1]))
      BUG();
```

Analysis Boosting

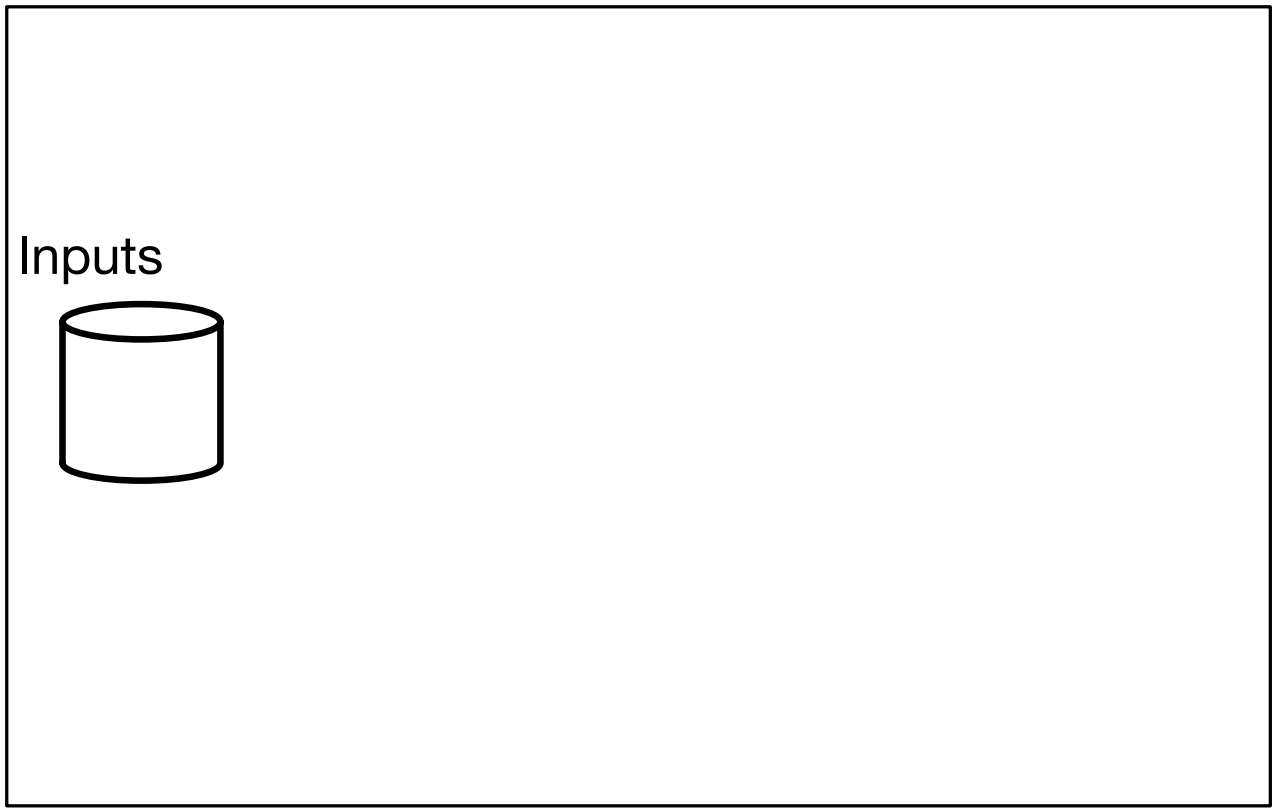
- How do you combine **existing** analysis tools?

Analysis Boosting

- How do you combine **existing** analysis tools?
- ‘Universal’ Knowledge: Inputs!

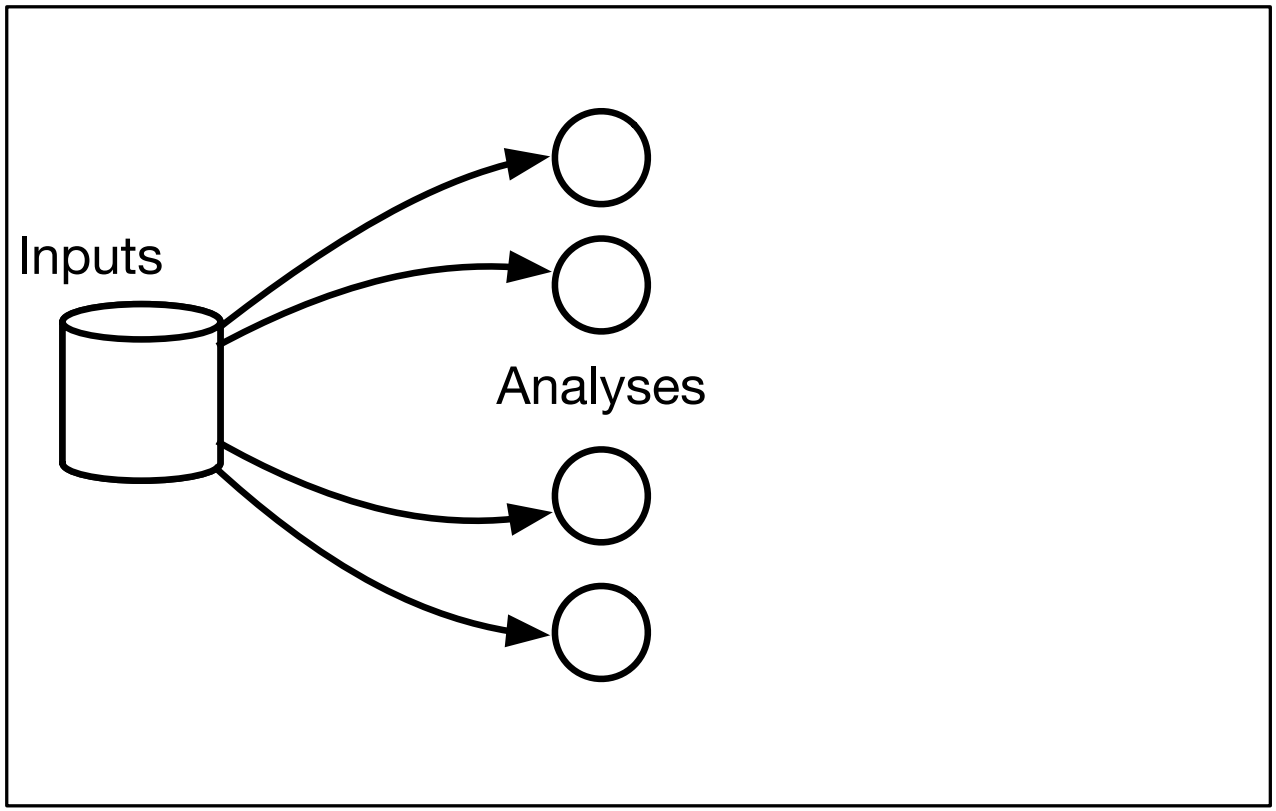
Analysis Boosting

- Inputs generated by one tool feed into all others



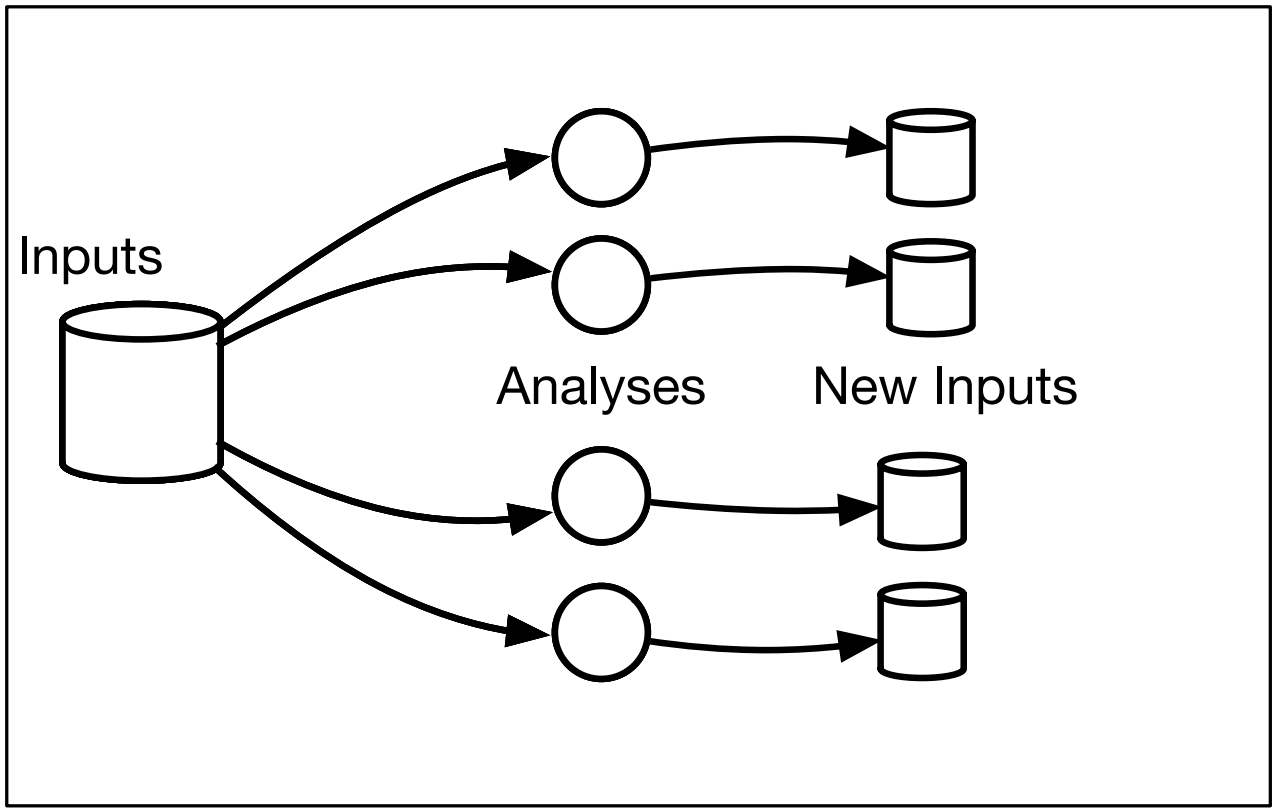
Analysis Boosting

- Inputs generated by one tool feed into all others



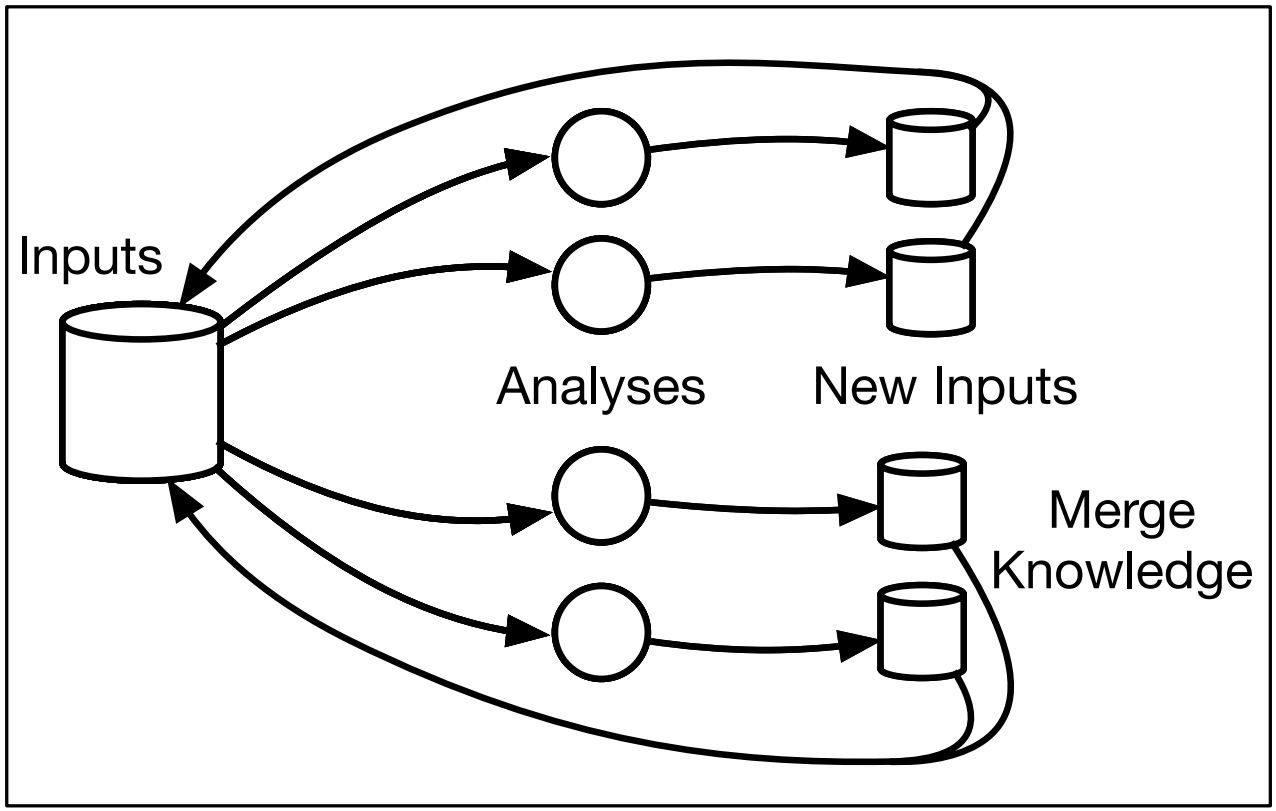
Analysis Boosting

- Inputs generated by one tool feed into all others



Analysis Boosting

- Inputs generated by one tool feed into all others



Analysis Boosting: MinSet

- Which inputs are “good”?
  - “AAAA…” vs. “\x7fELF...”

Analysis Boosting: MinSet

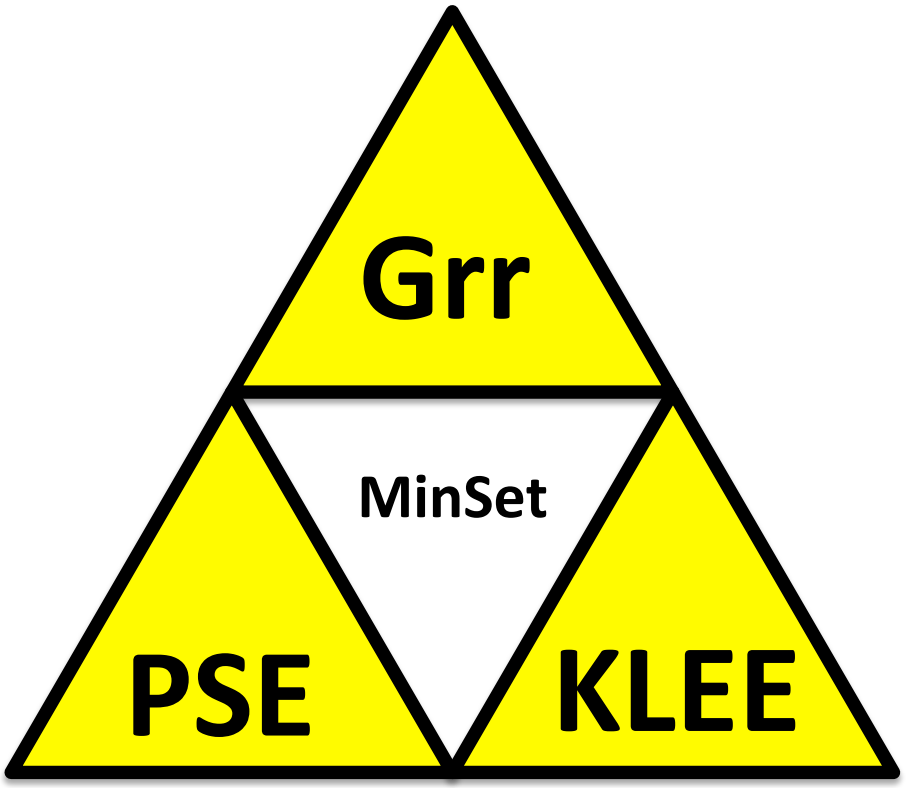
- Which inputs are “good”?
  - “AAAA…” vs. “\x7fELF...”
- MinSet: **Min**imum **Set** of Maximal Code Coverage

Analysis Boosting: MinSet

- Which inputs are “good”?
  - “AAAA…” vs. “\x7fELF...”
- MinSet: **Min**imum **Set** of Maximal Code Coverage
- Only keep inputs that generate new branch coverage.

Analysis Boosting

- It's like the Triforce.



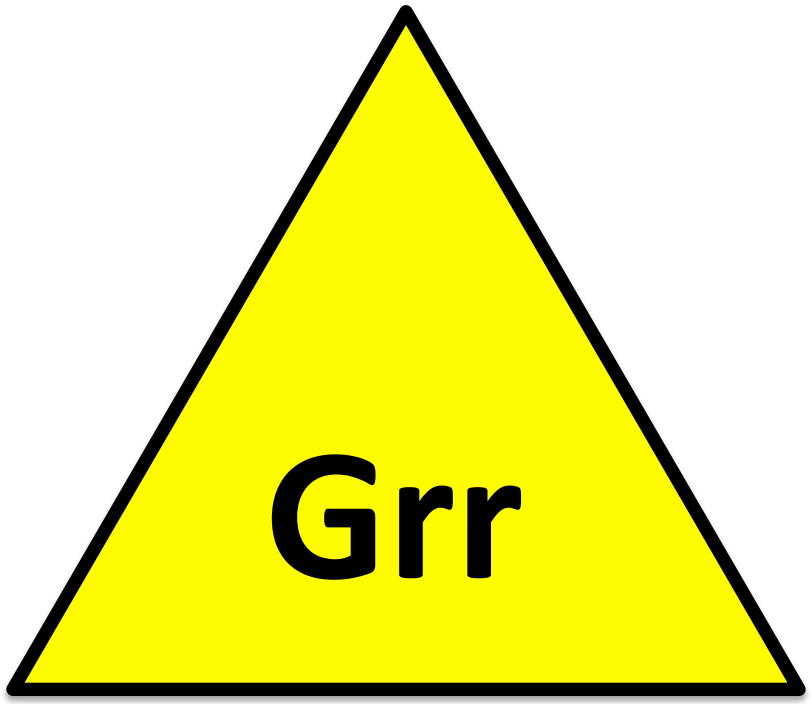
Analysis Boosting: Results

- Tools will cooperate to find bugs.
- A real crash history track:

klee/1/testcase_262069… =>
pysymemu/1/testcase_11f2b1…' =>
grr/1/**crashing**_testcase_1231f9…

Grr: A Faster Fuzzer

- Fuzzing is slow!
- Yes, really.



Why Fuzzing Is Slow

- Mutation Delay

Why Fuzzing Is Slow

- Mutation Delay
- Instrumentation Overhead

Why Fuzzing Is Slow

- Mutation Delay
- Instrumentation Overhead
- Startup Delay

Why Fuzzing Is Slow

- Mutation Delay
- Instrumentation Overhead
- Startup Delay
- Kernel Overhead

Grr: A Faster Fuzzer

- There is too much latency
- Most fuzzing time is **spent waiting for input**

Grr: A Faster Fuzzer

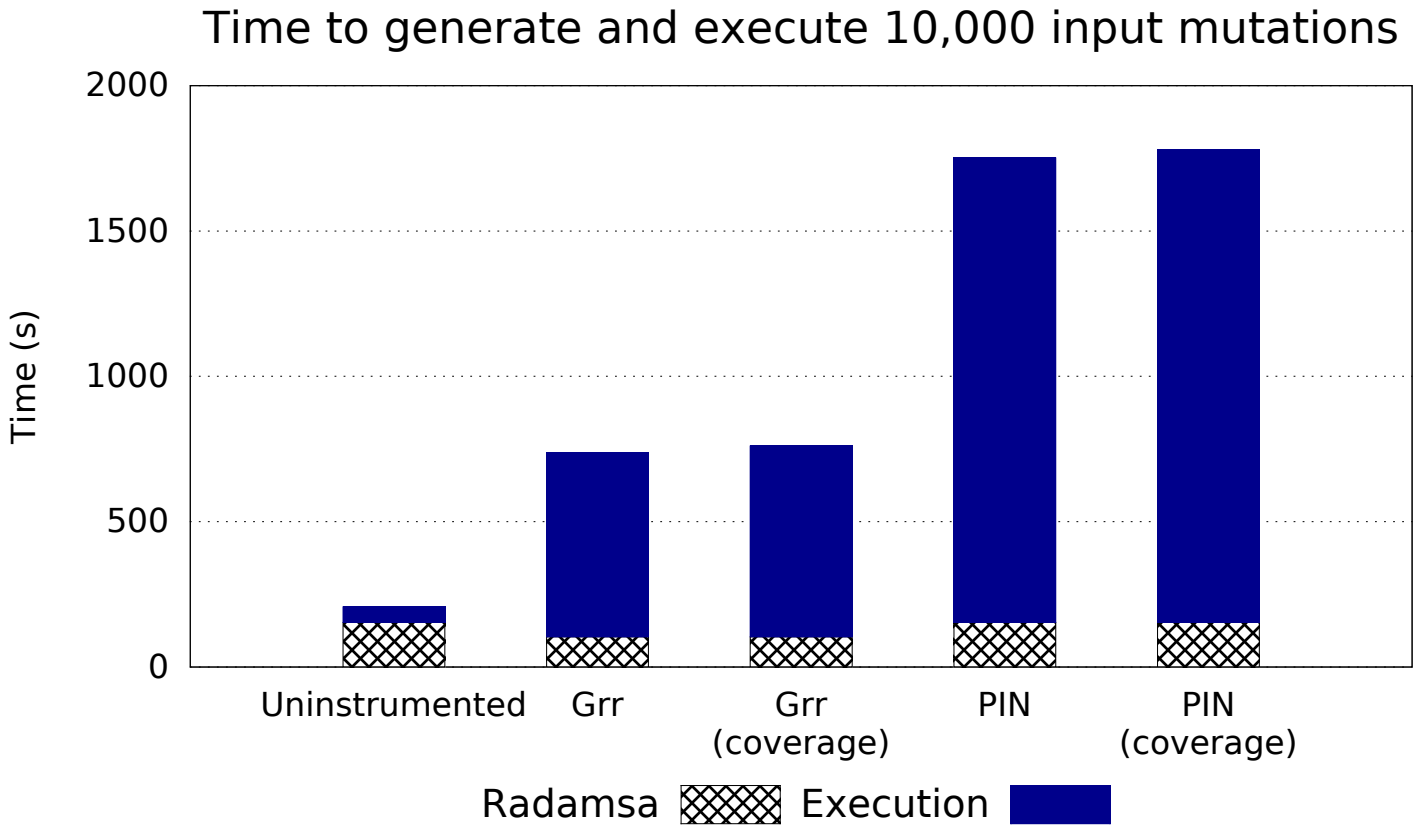
- Grr aims to minimize fuzzing latency
- Secret: Dynamic Binary Translation

Grr: A Faster Fuzzer

- Save and load snapshots
- Emulate system calls (no use of kernel)
- Inline code coverage
- Inline crash detection
- Mutate on per-syscall or whole file basis
- Re-use mutation engines (e.g. Radamsa)

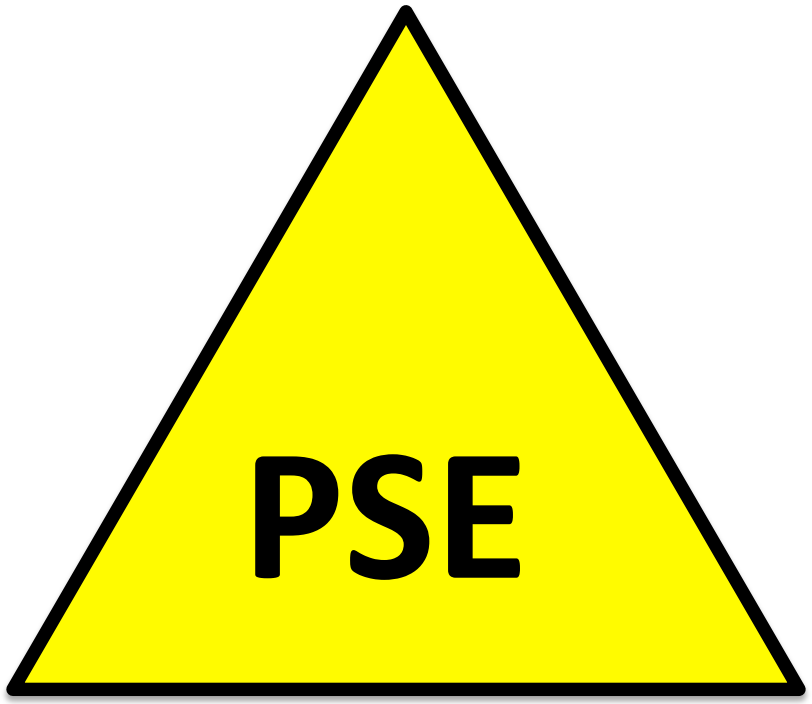
Grr: A Faster Fuzzer

- Extremely fast, provides free branch coverage and crash detection



PSE

- We thought it was great, so we hired the author



PSE

- Directly translates x86 to symbolic expressions
- Supports self-modifying code
- Pluggable CPU and OS models
- Saves state to disk
- Z3 solver backend

PSE

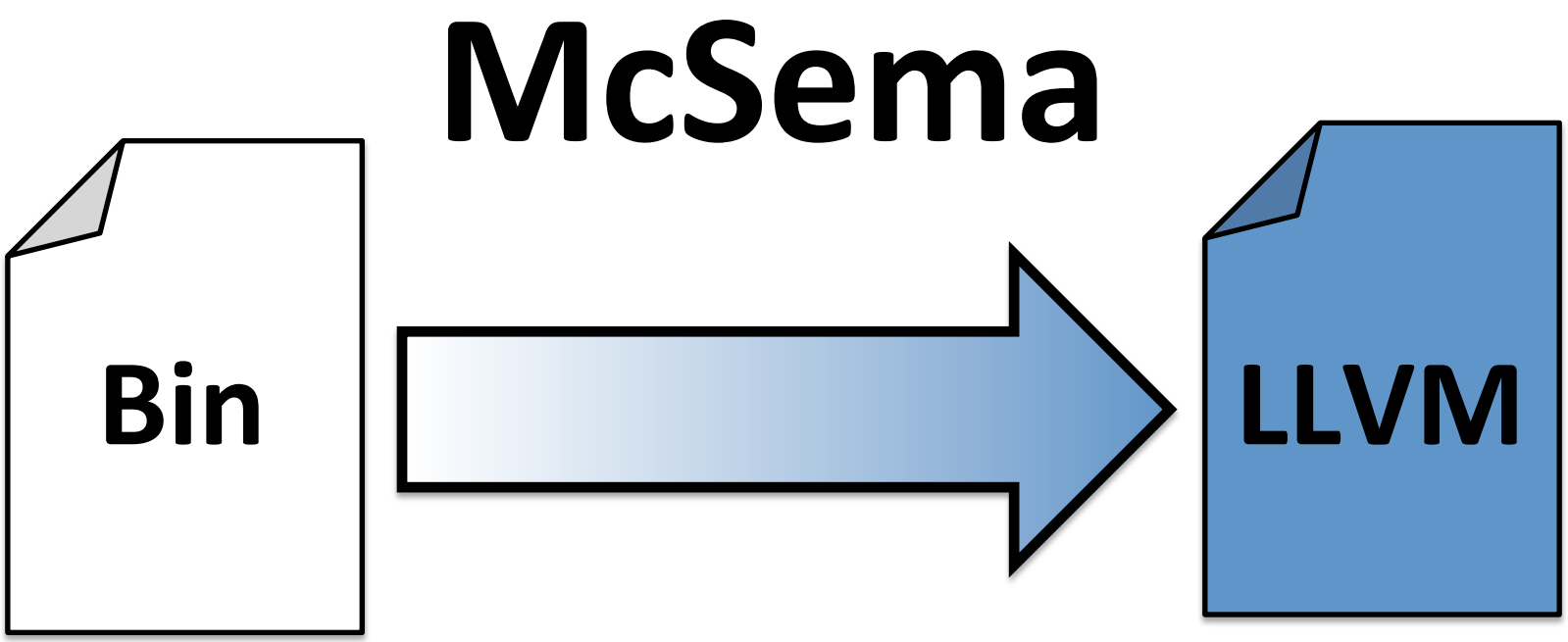
- Starts with initial concrete inputs
  - Useful for avoiding path explosion
- Starts with initial Grr snapshot
  - Also useful for avoiding path explosion

KLEE

- Well-known symbolic execution engine
- But this only works with source?!

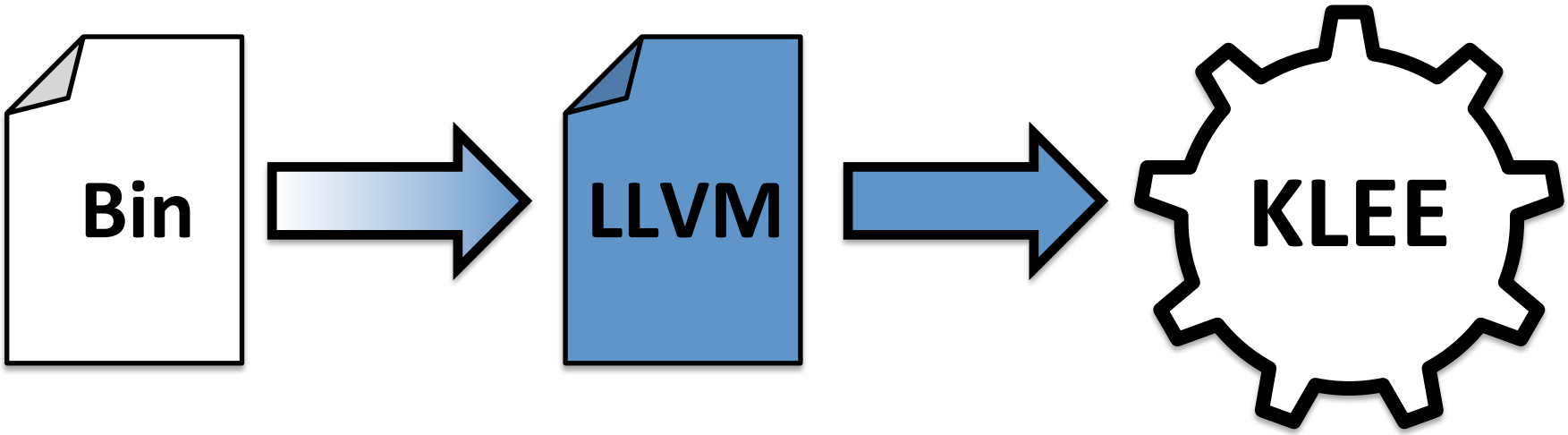
KLEE On Binaries

- Just Add...



KLEE On Binaries

- Yes, this actually works



The Second Scored Event

- Scored Event 2. April 2015.
- 2 Months to go.
- ToB Score…

The Second Scored Event

- Scored Event 2. April 2015.
- 2 Months to go.
- ToB Score:

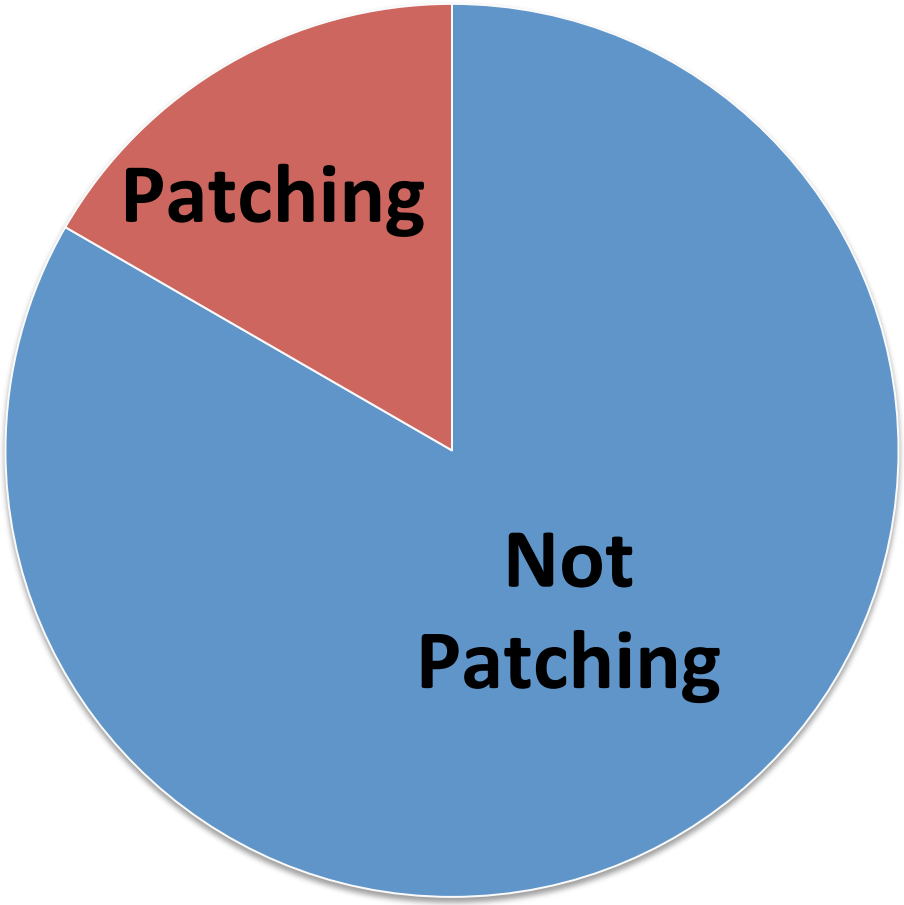
0

What is “Patching”

Patching

- Patching is for whitehats

Development Time Spent



Pwning

- Distributed system **finally works**
- We find bugs
- We are patching
- The patching is acceptable
- We are web scale

The Final Test

- CGC Qualifying Event. June 2015.
- What we worked towards for a year
- 24 hours of head to head competition
- No human intervention allowed

CQE: Operation

- 297 ‘c4.8xlarge’ Amazon EC2 Instances (almost all US capacity)
- 10,692 Cores
- 17,820 GiB of RAM
- 3 EC2 availability zones
- 232 TiB of disk
- 2.5 hours on phone with Amazon Support to set it all up

CQE: Operation

- We overprovisioned by about 15x
- Due to unknowns and paranoia
-  Spent ~$27K in 24 hours 
- Could have spent ~$1.8K 💵

CQE Results

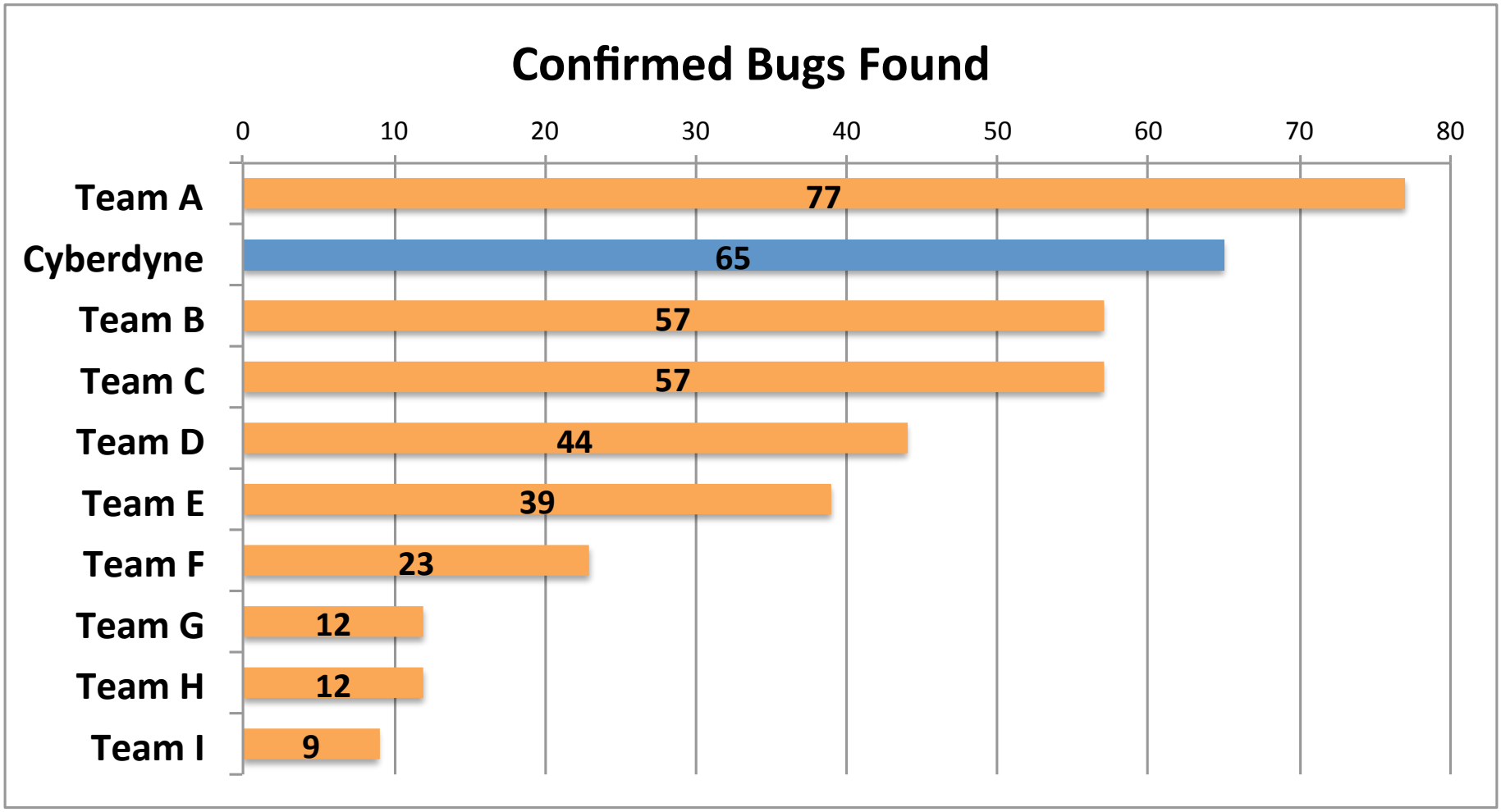
- At first, we thought we kicked ass
- Top 7 Qualify
- ToB got…

CQE Results

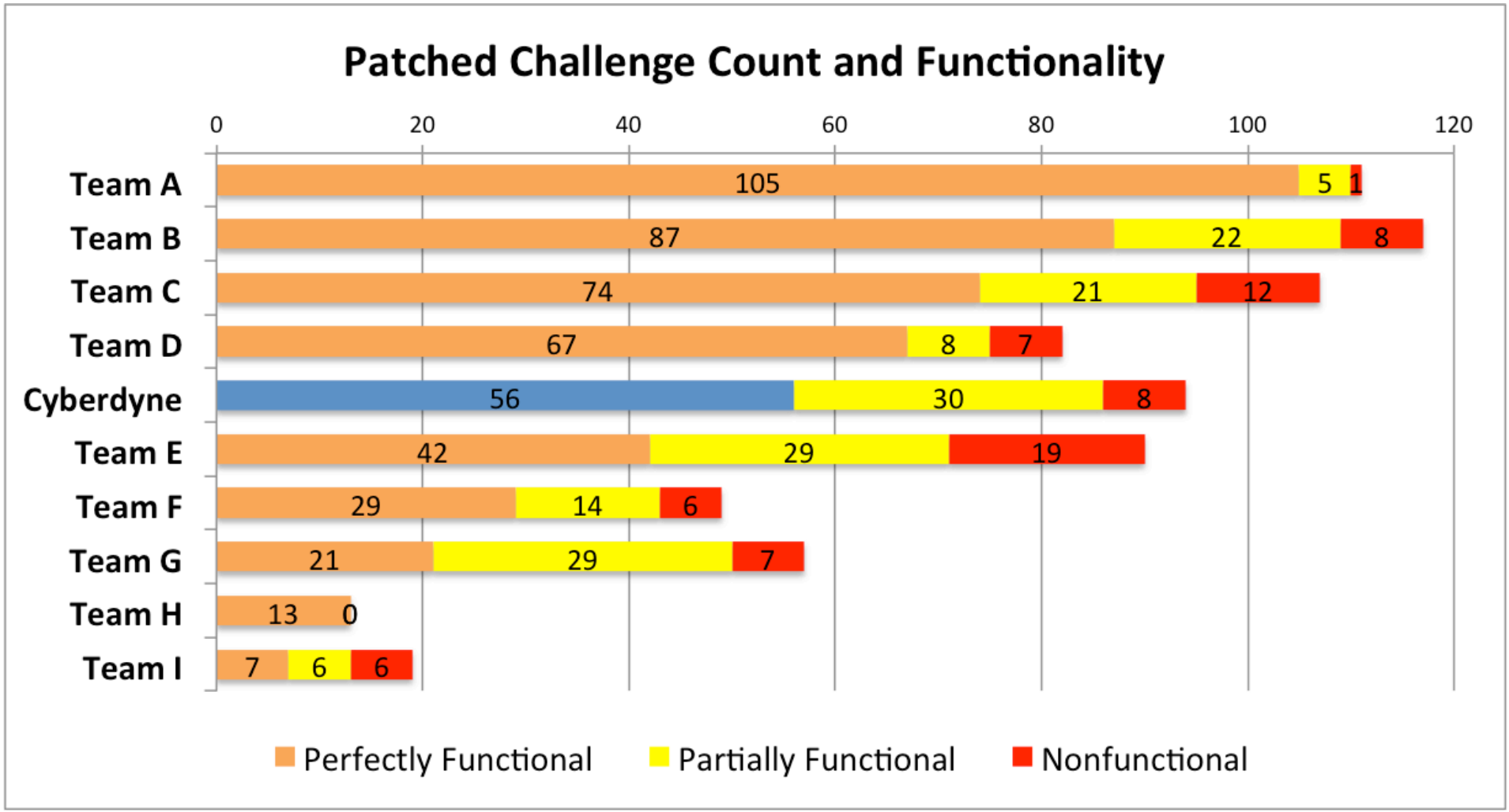
- At first, we thought we kicked ass
- Top 7 Qualify
- ToB got:

9th

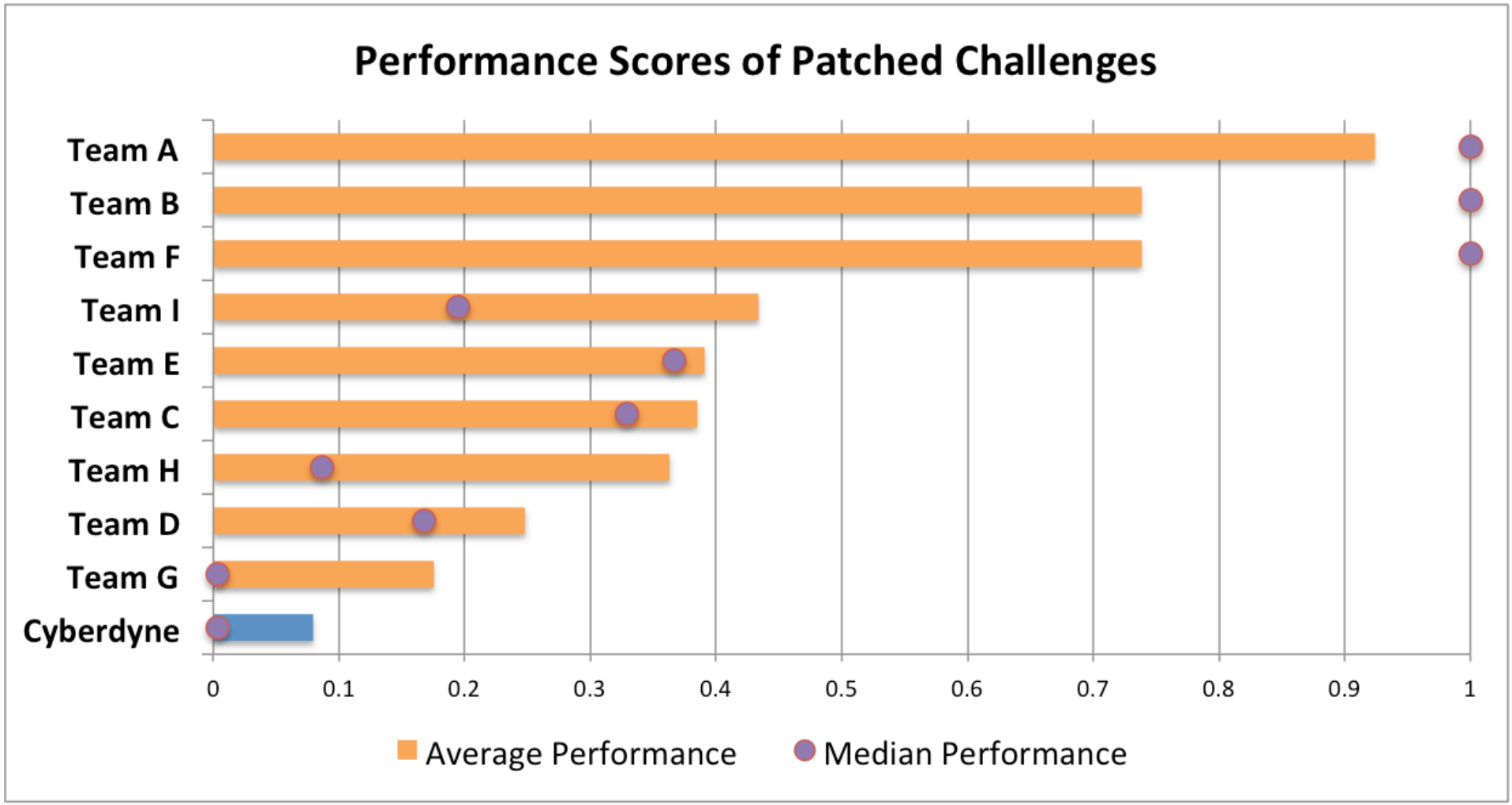
CQE Results: 2nd in Bugs Found



CQE Results: 5th in Patching



CQE Results: Last in Performance



CQE Aftermath

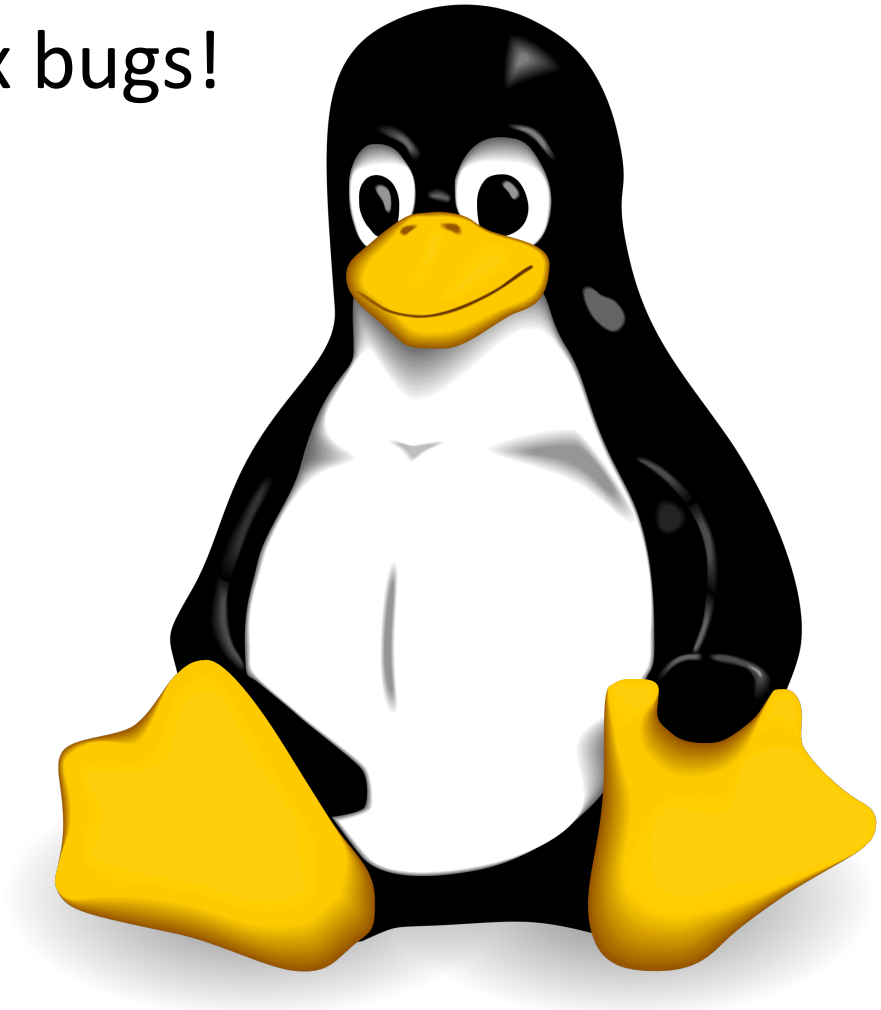
- Rules were clear
- We focused on the wrong things
- We failed
- … but others go on!

CGC Is Not Over!

- Cyber Grand Challenge Final Event
- When: Thursday, August 4, 2016 (5:00pm – 8:00pm PCT)
- Where: Paris Hotel and Conference Center, Las Vegas, NV
- Details: Free & open to the public

What Now?

- Find Linux bugs!



Linux Tests

- Build library to LLVM
- Create a libc wrapper
  - Libc
  - Filesystem
  - Environment Vars
  - Users

Linux Tests

- libotr
  - Treat it as black box, keep MACs and encryption.
  - No results ☹

Linux Tests

- libharfbuzz
  - Google already fuzzed it ☹
  - Cyberdyne could re-find those bugs 🙂

Linux/OSX Tests

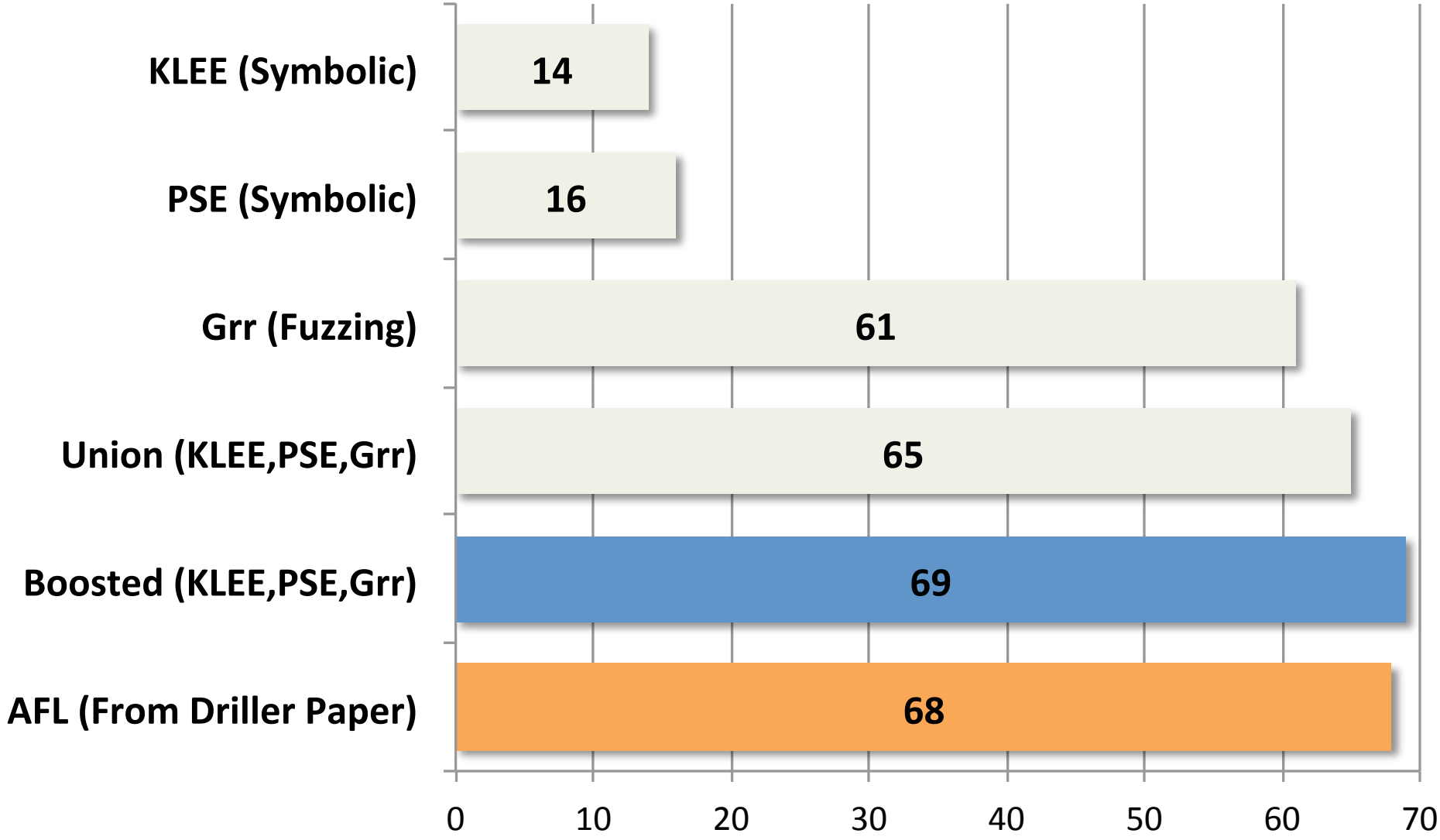
- libarchive

Linux/OSX Tests

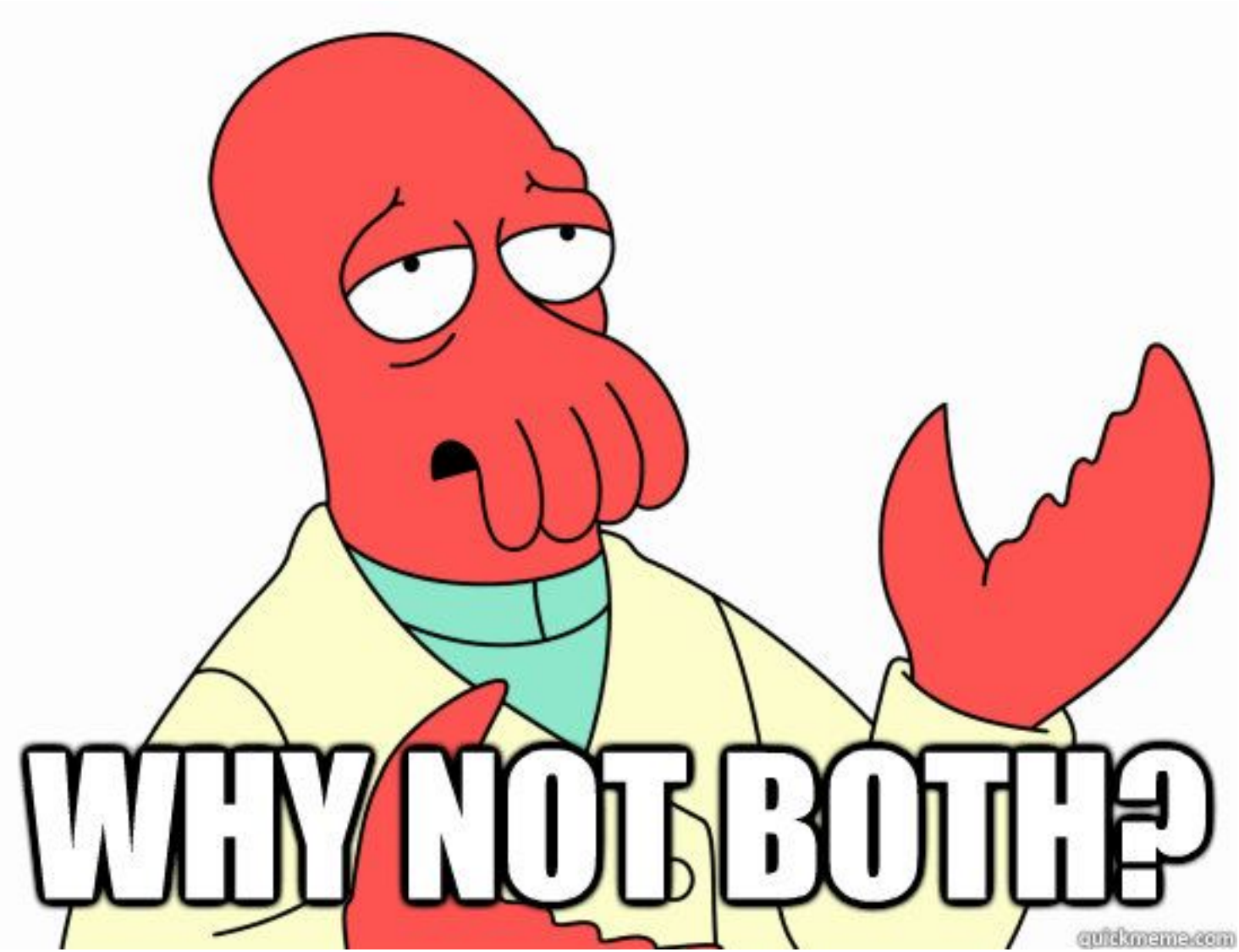
- libarchive


```
$ lldb -- ./tar -x -f untar_804c.bin
Process 2315 launched: './tar' (x86_64)
Process 2315 stopped
* thread #1: tid = 0x33fe2, 0x00007fff911f92b9 libarchive.
2.dylib ... stop reason = EXC_BAD_ACCESS (code=1,
address=0xff5f0800)
    frame #0: 0x00007fff911f92b9:
->  0x7fff911f92b9 <+148>: cmpb   $0x0, (%r13)
```

Bugs Found in CQE Binaries By Method



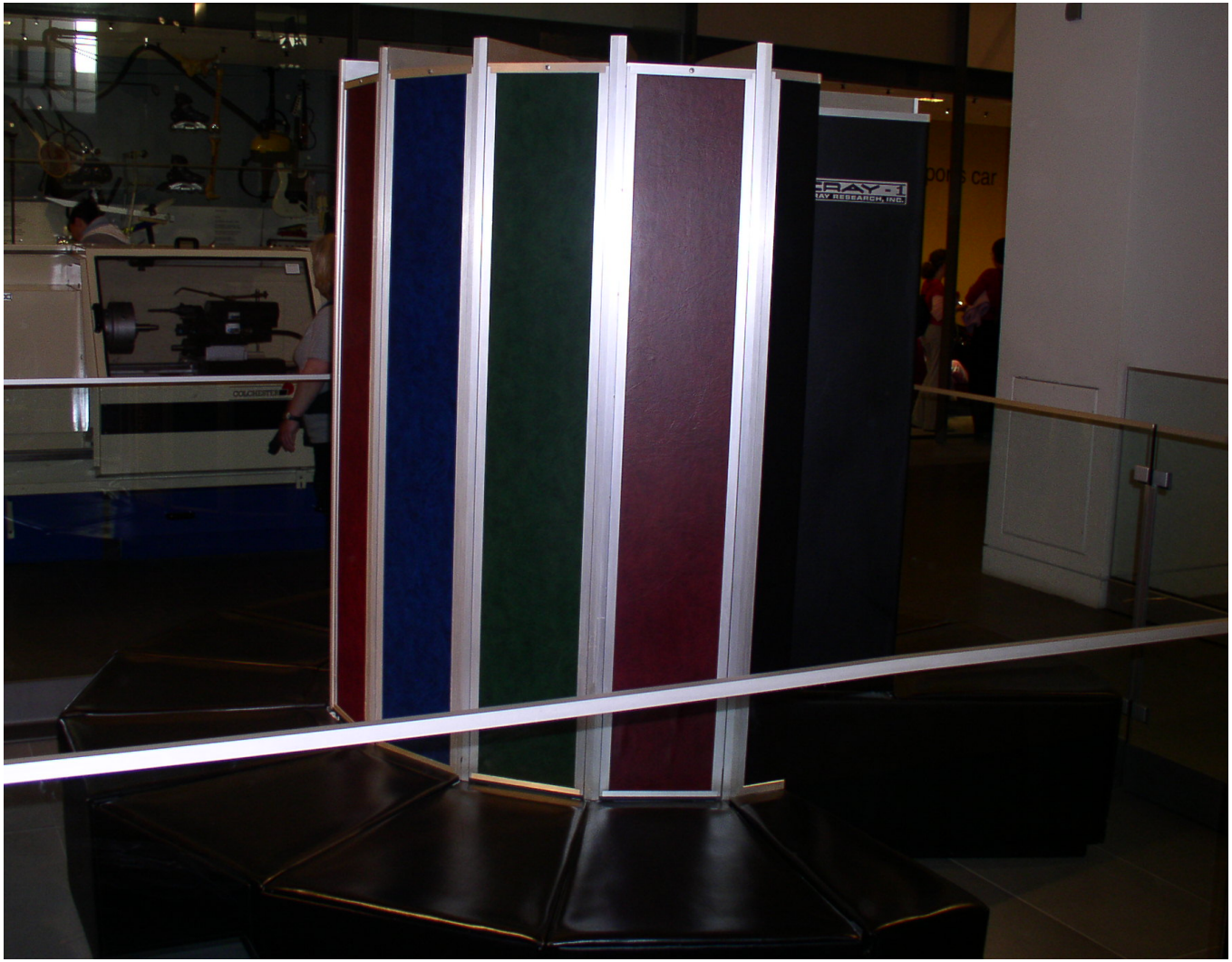
Cyberdyne or AFL or … ?



Conclusion

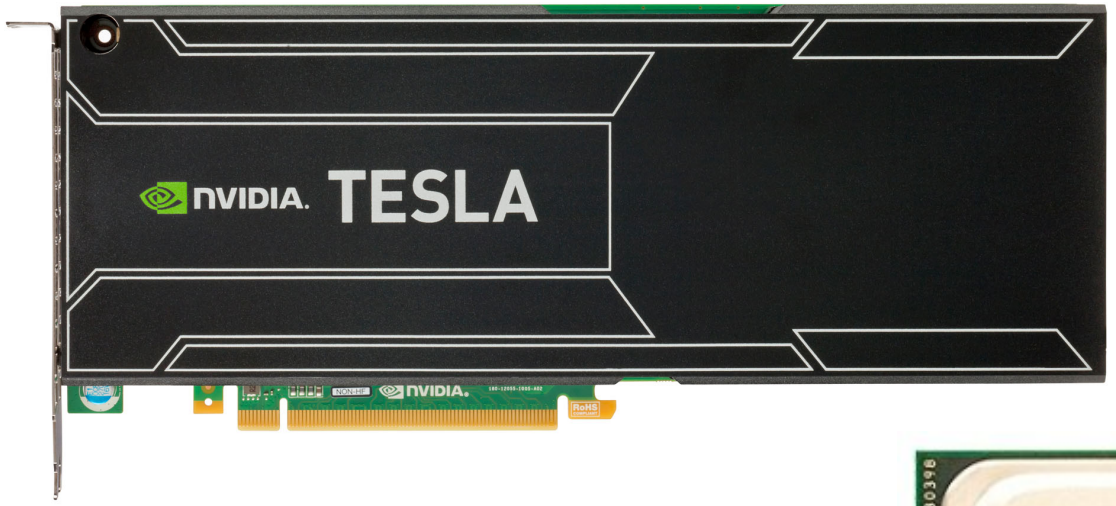
- We have been building tools wrong
- Small, cooperative tools are the future
- Especially when you have to distribute them
- Analogy time…

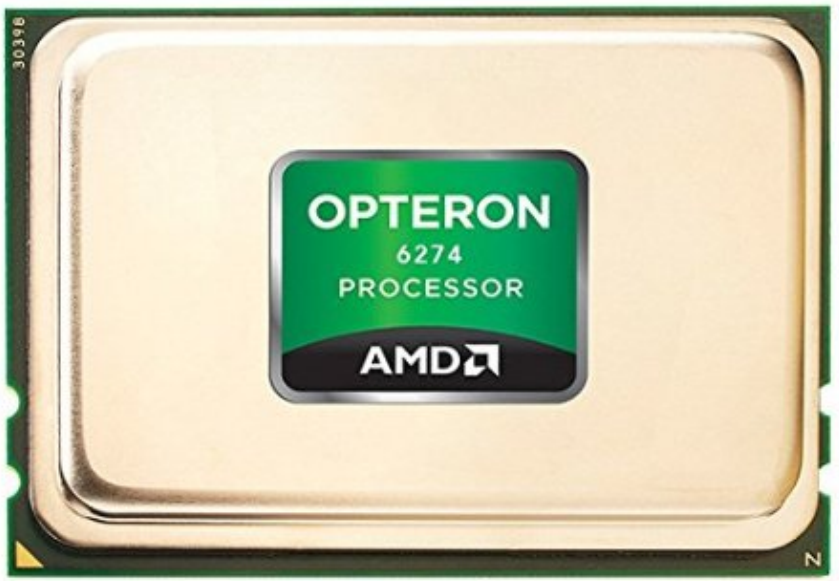
Bug Finding Tools Today: Cray-1



Bug Finding Tools Tomorrow:
Cray Titan

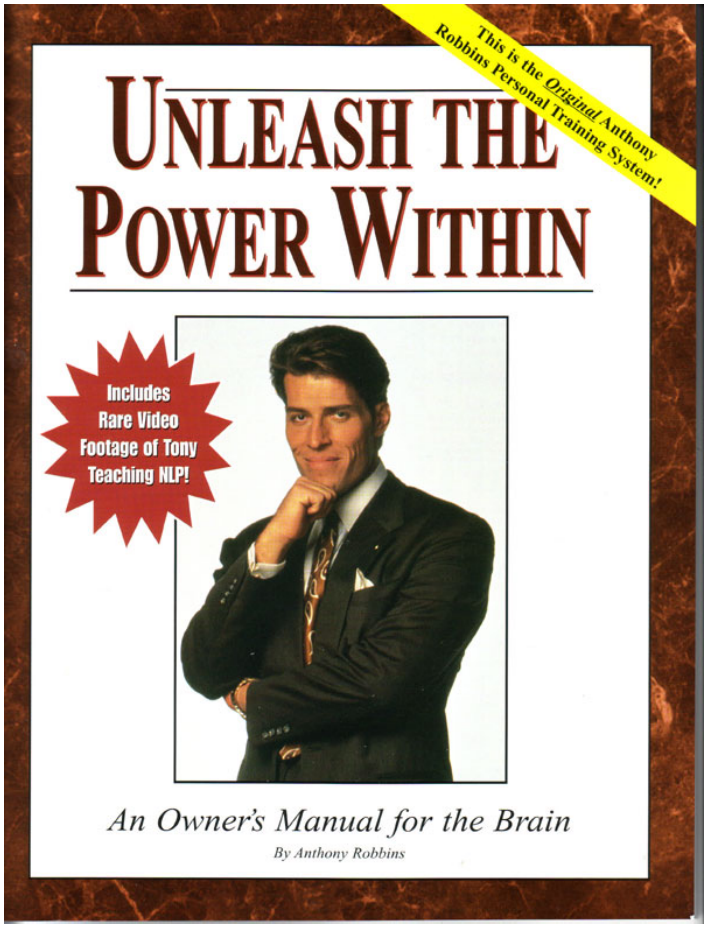
Bug Finding Tools Tomorrow: Cray Titan





Unleash The Power Within

- Publish machine readable analysis state!



Questions?

Contact Information:

artem@trailofbits.com
http://blog.trailofbits.com

Further Reading:

http://blog.trailofbits.com/2015/07/15/how-we-fared-in-the-cyber-grand-challenge/

http://blog.trailofbits.com/2016/01/13/hacking-for-charity-automated-bug-finding-in-libotr/

https://github.com/trailofbits/mcsema

https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/walker