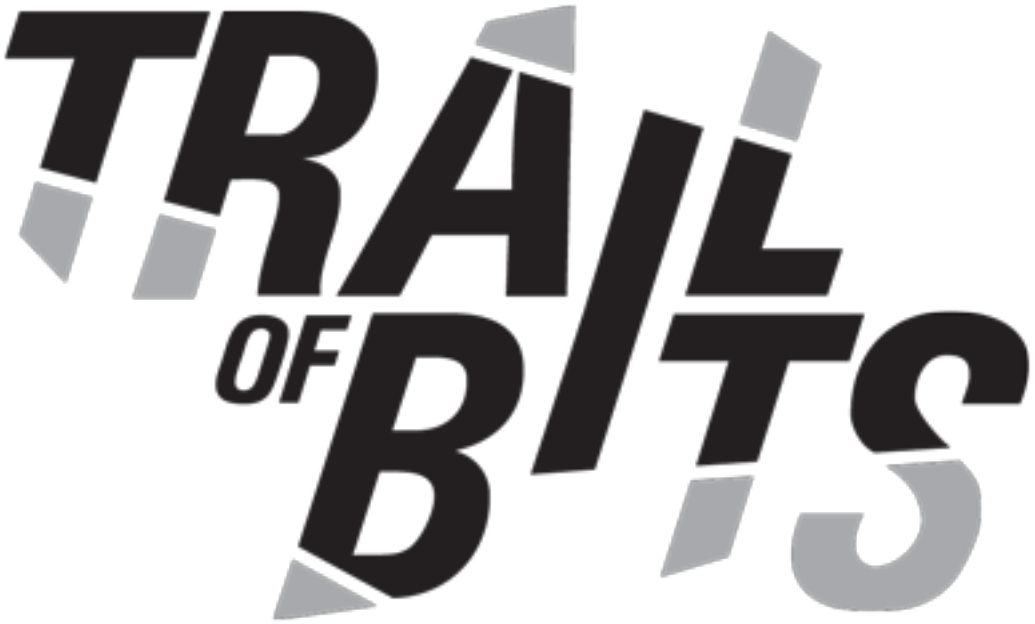
New Perspectives on Automated Vulnerability Discovery

Artem Dinaburg

Trail of Bits

About Me



About This Talk

- Why are automated vulnerability discovery tools rarely used?

About This Talk

- Why are automated vulnerability discovery tools rarely used?
- Changing development methodology to make these tools more accessible.

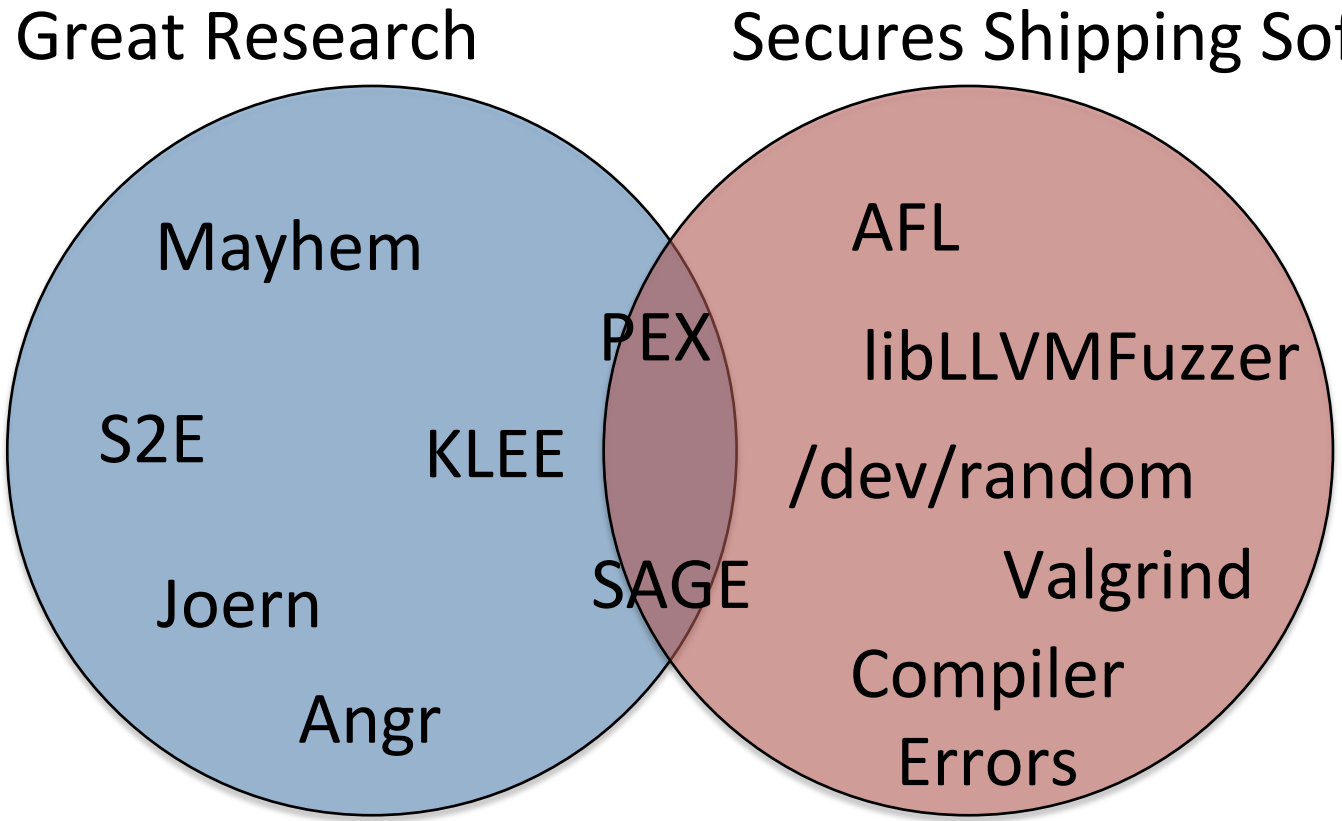
About This Talk

- Why are automated vulnerability discovery tools rarely used?
- Changing development methodology to make these tools more accessible.
- Our experience using this methodology to compete in DARPA's Cyber Grand Challenge.

About This Talk

- Why are automated vulnerability discovery tools rarely used?
- Changing development methodology to make these tools more accessible.
- Our experience using this methodology to compete in DARPA's Cyber Grand Challenge.
- Future research

Bug Finding Tool Disparity



Automated Vulnerability Discovery (now)

- It works!

Automated Vulnerability Discovery (now)

- It works!
- … If you are a part of the team that developed it.

Automated Vulnerability Discovery (now)

- It works!
- ... If you are a part of the team that developed it.
- ... And can firmly grasp the concepts

Automated Vulnerability Discovery (now)

- It works!
- ... If you are a part of the team that developed it.
- ... And can firmly grasp the concepts
- ... For the select few (<0.01%?)

Automated Vulnerability Discovery (goals)

- Lets expand tool use to the 1%
  - Popular open source projects
  - Commercial consumer facing software
- Make your research more impactful!

Why the 1%?

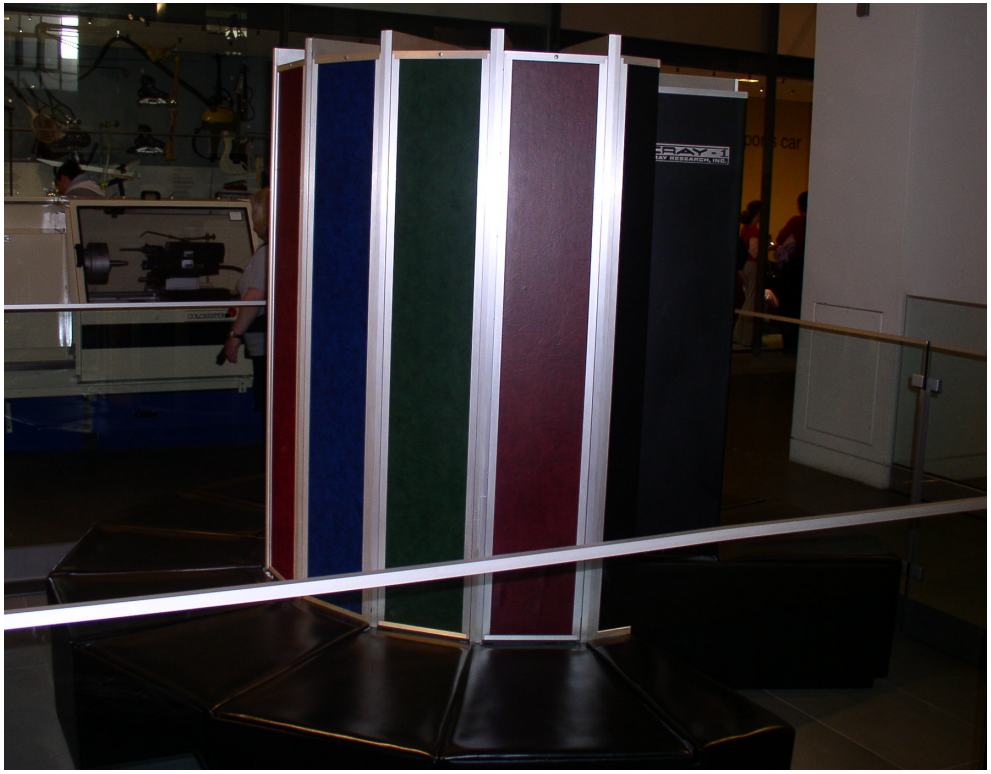
- Rough estimate of how many developers care.
- 99% use the “ostrich” approach to security.
  - This is fine. Really.

Barriers to Adoption

- Security Tool Adoption Barriers
  - Popularity (e.g. other people use them)
  - Solves real problems
  - Exposure (e.g. blogs, media)
- Not Barriers
  - Easy to use (just can't be absurdly hard!)

An Analogy…

- Automated Vulnerability Discovery Today



An Analogy…

- What Automated Vulnerability Discovery Should be

Better Development Methodology

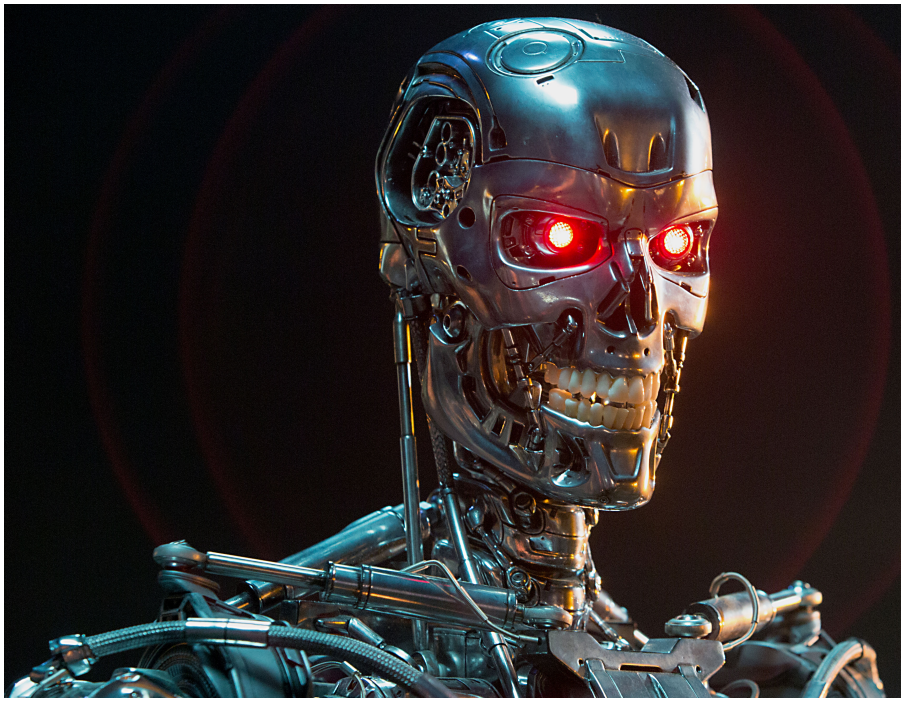
- Simple re-usable parts
  - Why are we re-writing all the things?
- Do one thing well
- Communicate
  - Common data interchange format!

Successful Models

- LLVM!
- Many (simple) tools
- Common interchange format (bitcode)
- Not easy, but not absurdly hard.

Case Study: Cyberdyne

- Our entry into the Cyber Grand Challenge.
- Composed of multiple communicating analyses.
- Evil corporation from the Terminator franchise.

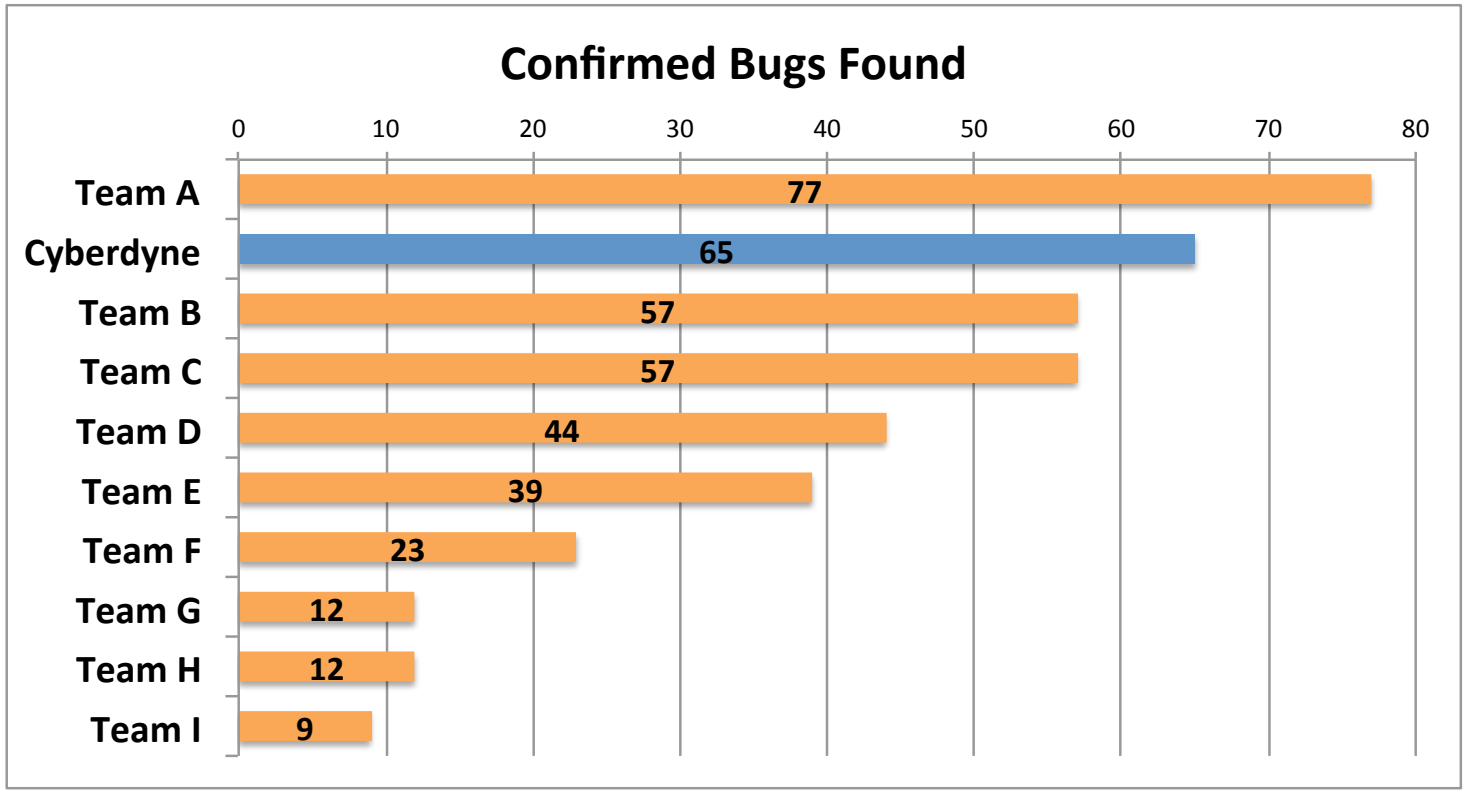


What is CGC?

- Competition to automate vulnerability discovery and patching
- In binary-only software
- Simplified OS
- Realistic example binaries
- Qualification Round, Final Round

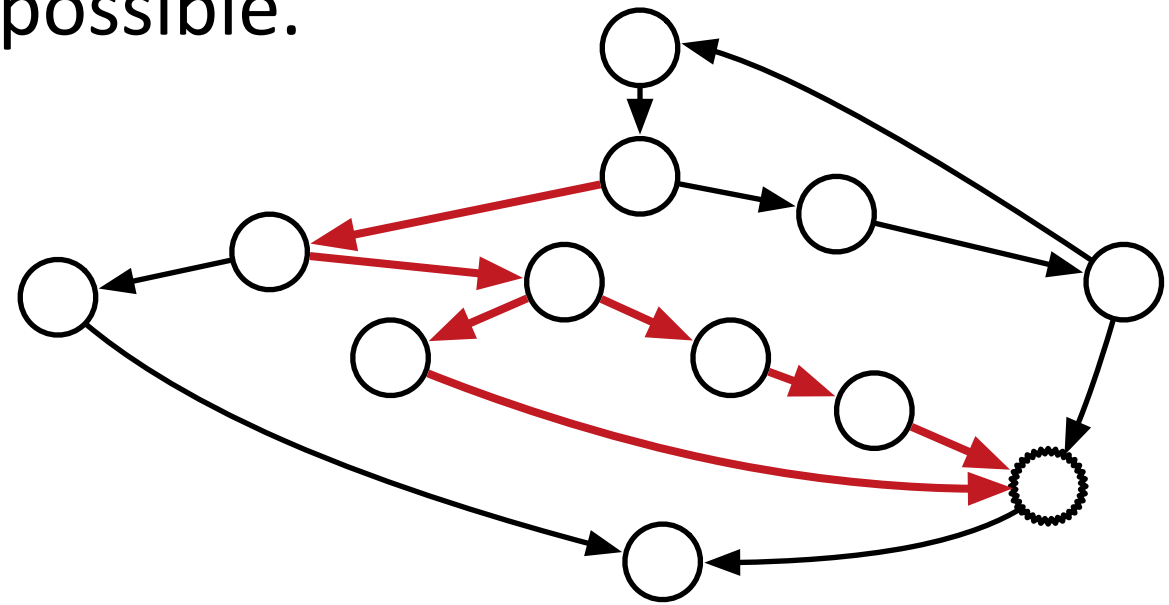
How Cyberdyne Fared

How Cyberdyne Fared



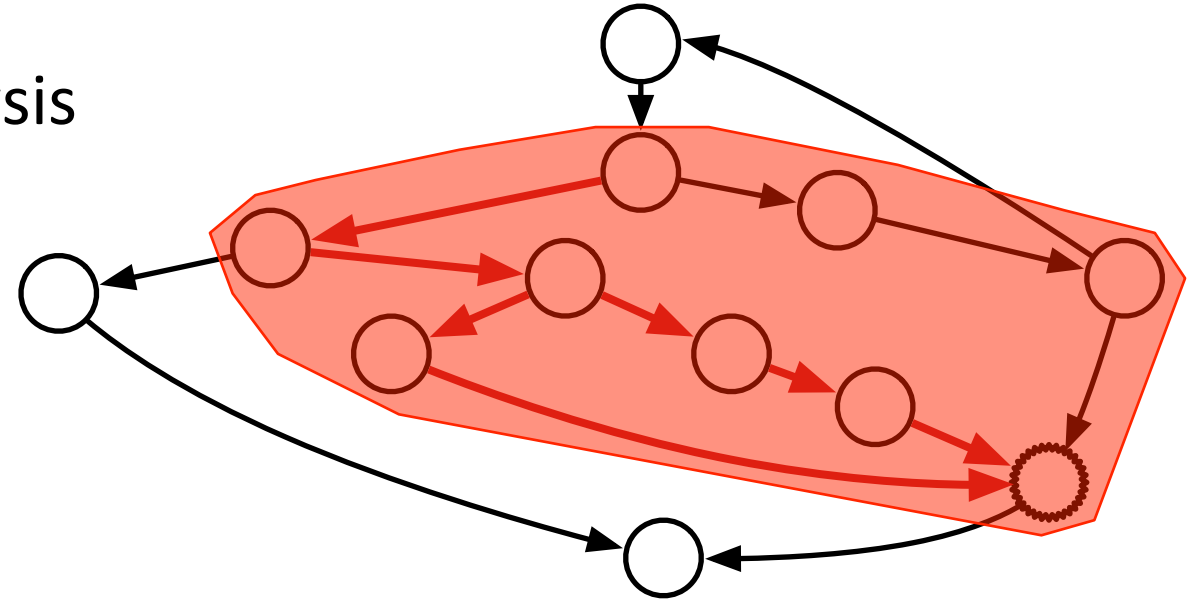
Vulnerability Discovery Theory

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



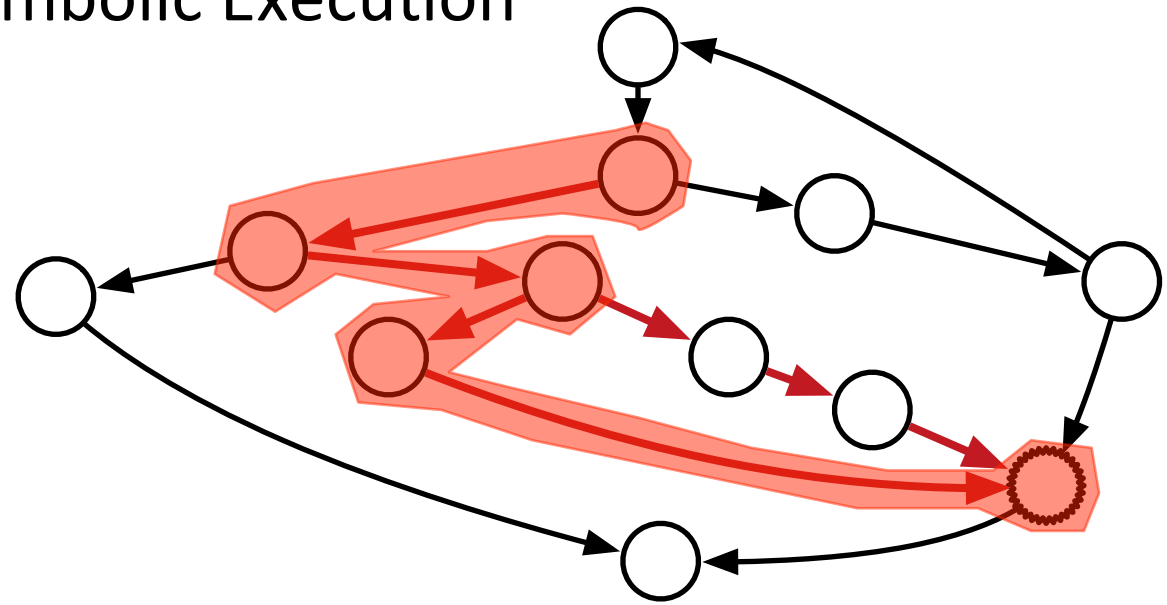
Vulnerability Discovery Theory

- Over Approximate Analyses
  - Points To
  - Alias Analysis



Vulnerability Discovery Theory

- Under Approximate Analyses
  - Fuzzing, Symbolic Execution



Under-Approximate Analyses: Roadblocks

Hard For Fuzzing, Easy for Symbolic Execution

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

Under-Approximate Analyses: 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 tools 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**

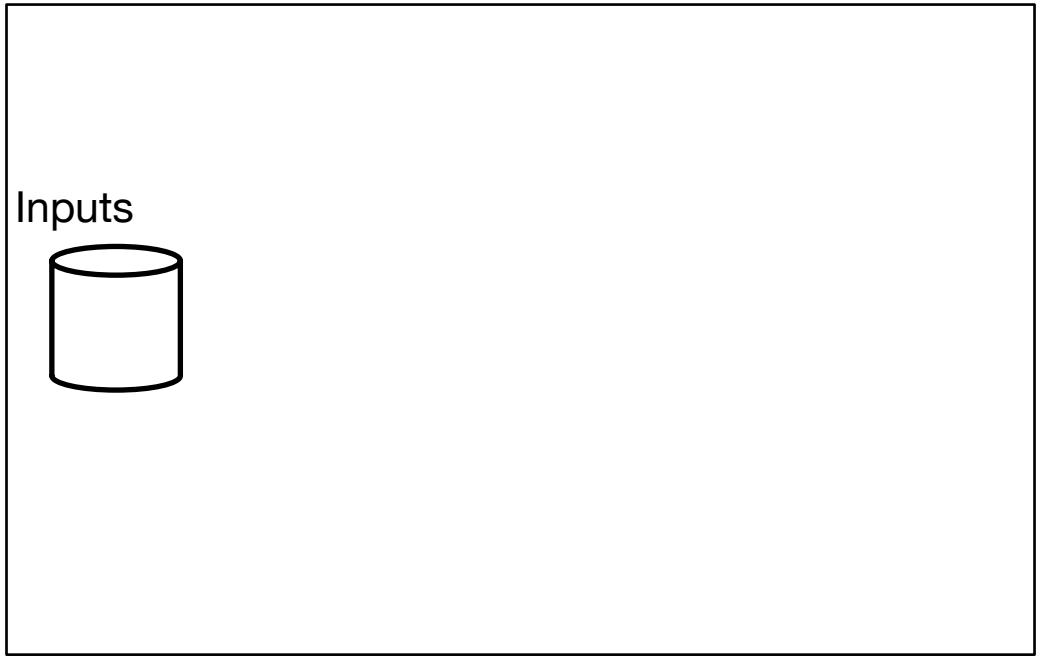
```
if(input[0] == 0xBADF00D)
  if(hash(input[0])
    == hash(input[1]))
      BUG();
```

Analysis Boosting: Communication

- Program Inputs!?
  - Convenient
  - Universal
  - Lame
- We can, and should, do better!

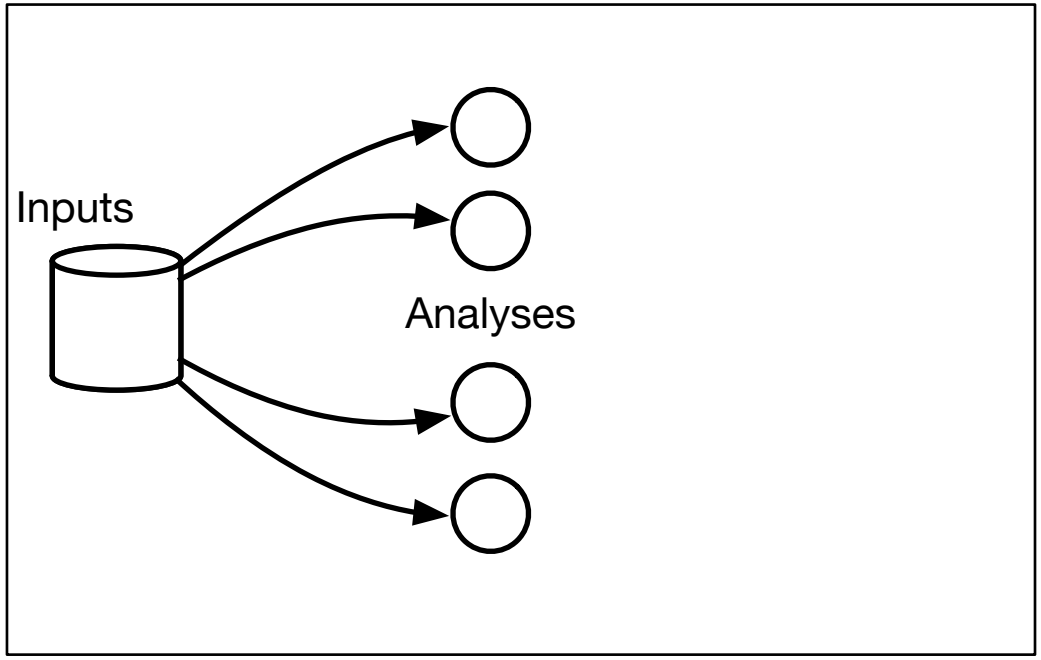
Analysis Boosting: Implementation

- Multiple tools communicating via program inputs



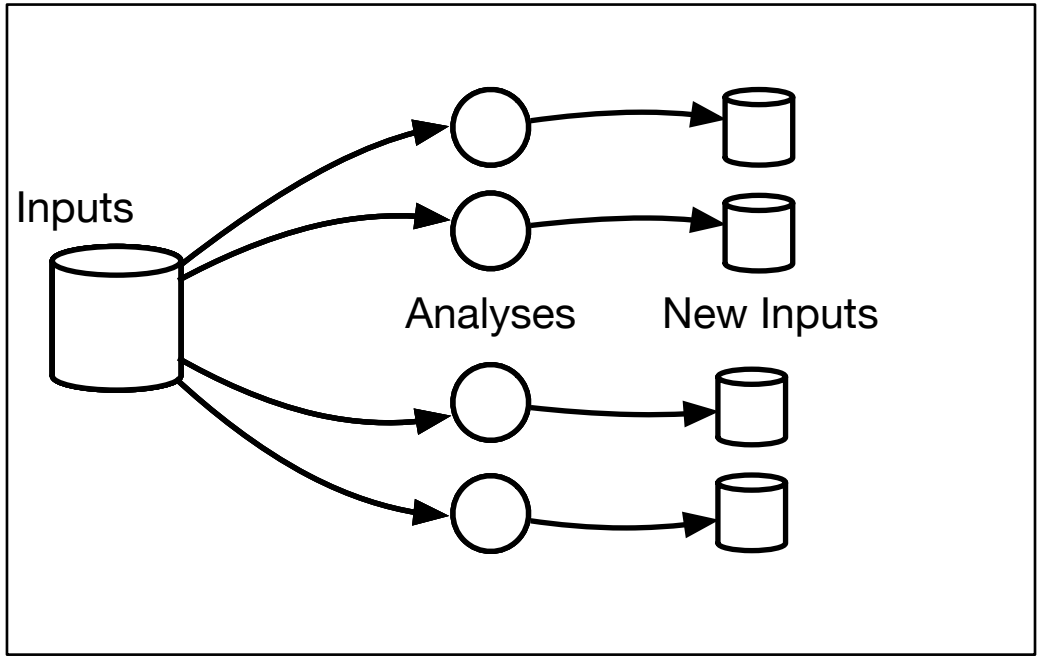
Analysis Boosting: Implementation

- Multiple tools communicating via program inputs



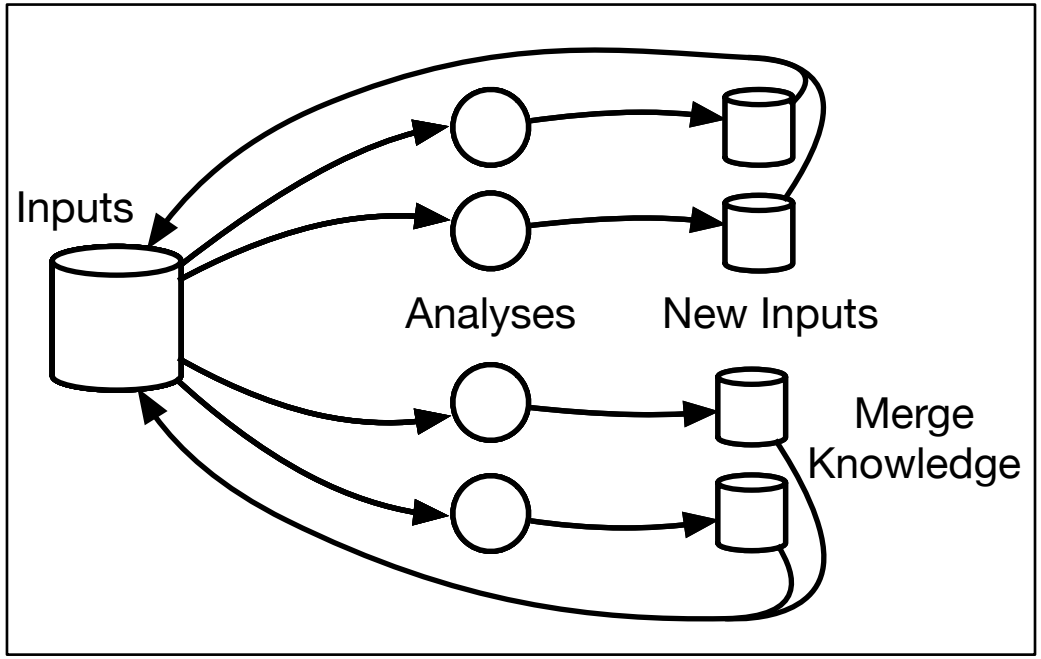
Analysis Boosting: Implementation

- Multiple tools communicating via program inputs



Analysis Boosting: Implementation

- Multiple tools communicating via program inputs

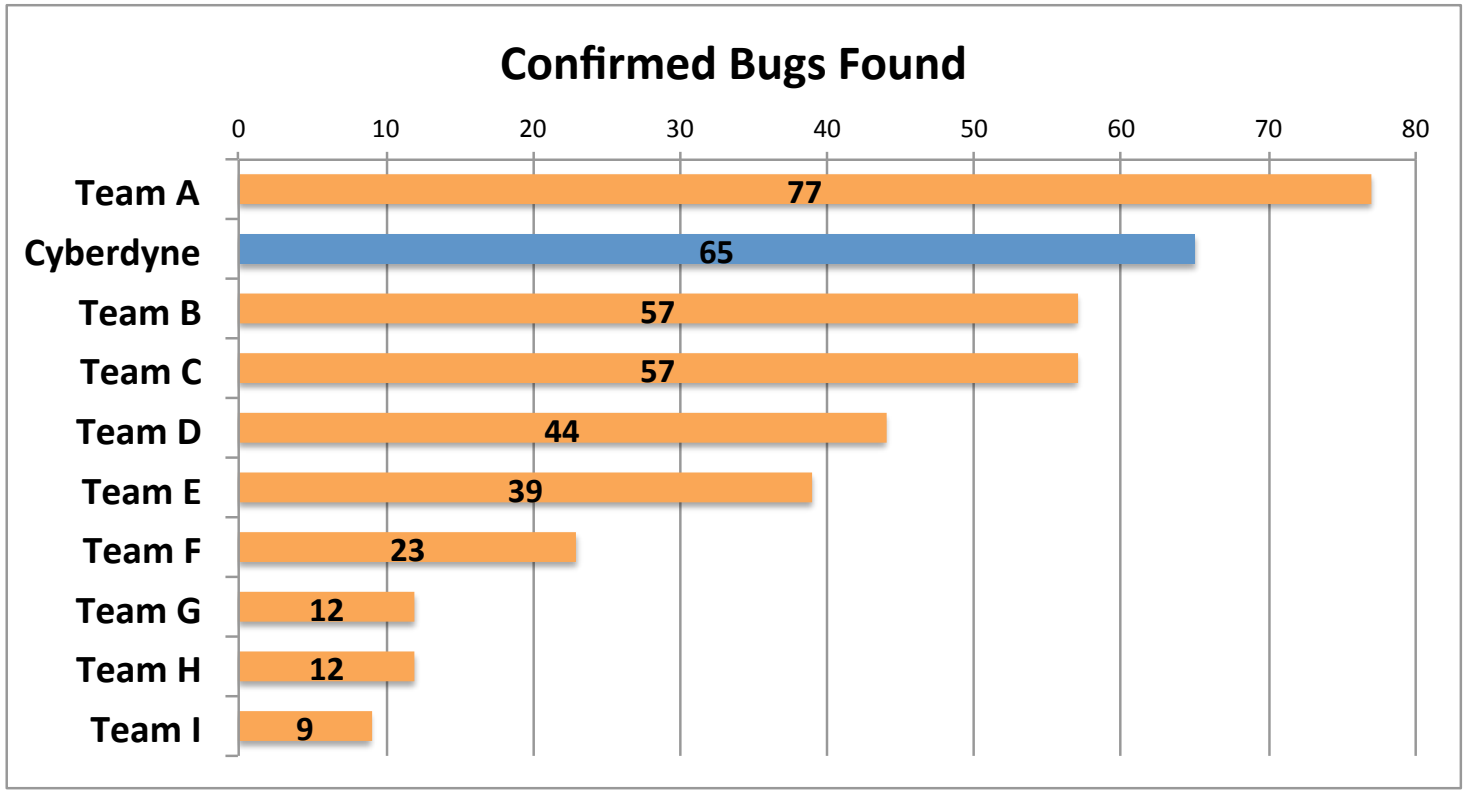


Analysis Boosting: It Works!

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

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

How Cyberdyne Fared (reminder)



Analysis Boosting: It Scaled

- This is ~15x overprovisioned, but we were paranoid.
  - 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

Analysis Boosting: Extensible

- New tools = new capability
- Linux libraries
  - libotr
  - libharfbuzz
  - libarchive
  - libwebp

Analysis Boosting: Future

- Sharing only program inputs is stupid
  - Throws away information
- We need better a data interchange method
  - Graph Database?

Conclusion

- Latest research developments aren't used to secure real software.
  - Complex
  - Monolithic
  - Brittle
- We can, and should, change that.

Conclusion

- Simple, communicating tools work
  - More accessible
  - Equivalent effectiveness
  - Easier to distribute
  - Easier to maintain and debug

Conclusion

- Lets build better analysis tools.
- Stop reinventing the wheel.
- Lets create a good analysis information interchange format.

Questions?

- Contact Information:
  - artem@trailofbits.com
  - http://blog.trailofbits.com