Build It Break It Fix It

Andrew Ruef

Who We Are

- builditbreakit.org
  - Dr. Michael Hicks, James Parker, and Andrew Ruef
  - University of Maryland
  - NSF funded, corporate sponsors prize money
- Trail of Bits
  - Andrew Ruef

What's the motivation?

- People love CTFs
- CTFs don't reflect reality
  - Reality isn't fun
- CTFs emphasize "breaking" / offensive
- Defensive CTFs are kind of a joke

Can we do better?

- We want a competition that:
  - Rewards people for finding bugs **and** making things
  - Is fun
- We want to focus more broadly than:
  - Rapid-fire IDA jam sessions
  - Patching and "compliance"
- Also, could we learn something about the world?

What's our idea?

- A contest where contestants
  - **Build** some secure software according to a specification
  - **Break** the software written by other contestants
  - **Fix** the bugs found in their software by other teams
- Organizers provide the specification
- Spread the contest over three weekends
- Announce two winners, one for best software, one for most bugs found

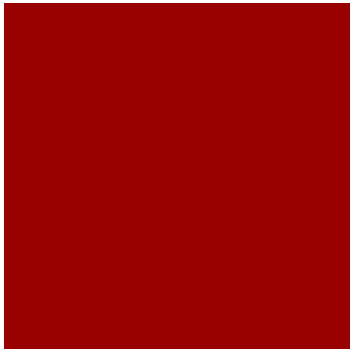
A "naturally occurring" CTF

- CTF problems are "canned", some very clever person created the problem wholesale
- "Bug" could also have a rich meaning based on the specification
  - If specification is "alarm system", "bugs" could allow undetected entry
  - More complicated and more meaningful than "look in IDA, get a shell"

Challenge specifications

- Needs to be at least a little fun
- Have high and low level security properties
  - Writing in Java or Python should not win by default
- Judge implementations on both correctness and performance
- Capable of unambiguously testing features
- Should be somewhat complicated, but doable in 72 hours

Fall 2014, alarm system

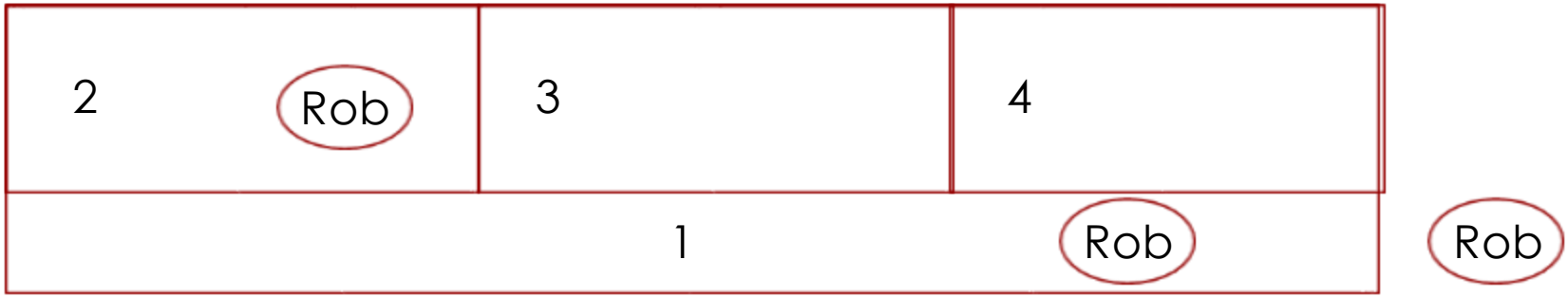


- Two programs, **logappend**, **logread**, manipulate a secure log file to either add events or query events
- Both programs authenticate to each other via a single shared symmetric key
- Programs that run faster are better
- Smaller log file size is better

Alarm system detail

- Series of rooms with numeric identifiers
- Individuals with alpha names
- Individuals can take an action at a timestep
  - Enter a room
  - Leave a room
- Individuals cannot do nonsensical things

Alarm system operation



logappend –T 1 –A –E Rob logfile

logappend –T 2 –A –E Rob –R 1 logfile

logappend –T 3 –L –E Rob –R 1 logfile

logappend –T 4 –A –E Rob –R 2 logfile

Build-It Exit Requirements

- Three categories of tests
  - Performance
  - Core
  - Optional
- An implementation "qualifies" if it passes all "core" tests
- Some features are "optional"

Performance

- Performance tests measure the efficiency of the application in space and time
- Time-based performance tests consider how long the application takes to run
- Space-based performance tests consider the size of the output file

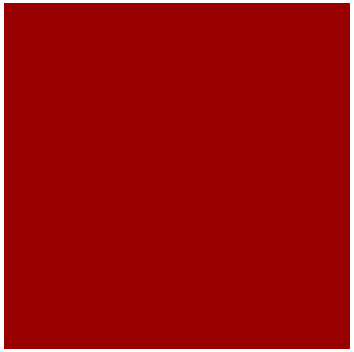
Three different types of bugs

- **Correctness** – The program didn't meet some part of the specification, or crashes
- **Integrity** – The log can be modified to attest to a false fact
- **Confidentiality** – The log can be analyzed to determine a protected fact
- We can automatically judge correctness and integrity bugs
- Integrity, confidentiality, and a correctness bug that produces a crash are counted as **exploits**

Infrastructure

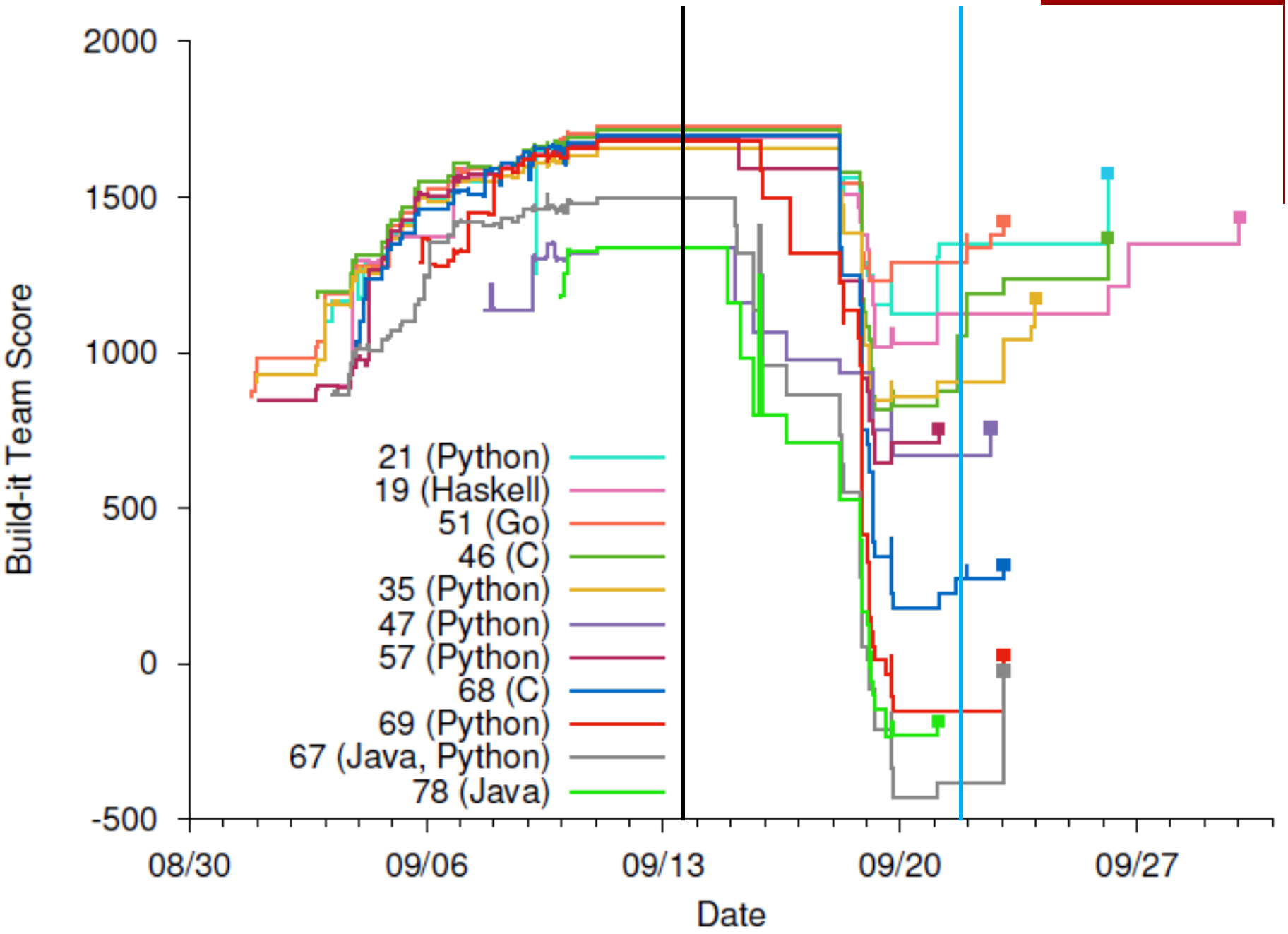
- This is still a hacking competition, it would be nice to not be compromised by our contestants
- Interface with contestants
  - A Haskell webapp
- Run contestant code
  - An EC2 backend to run every test in its own container

What were the results?

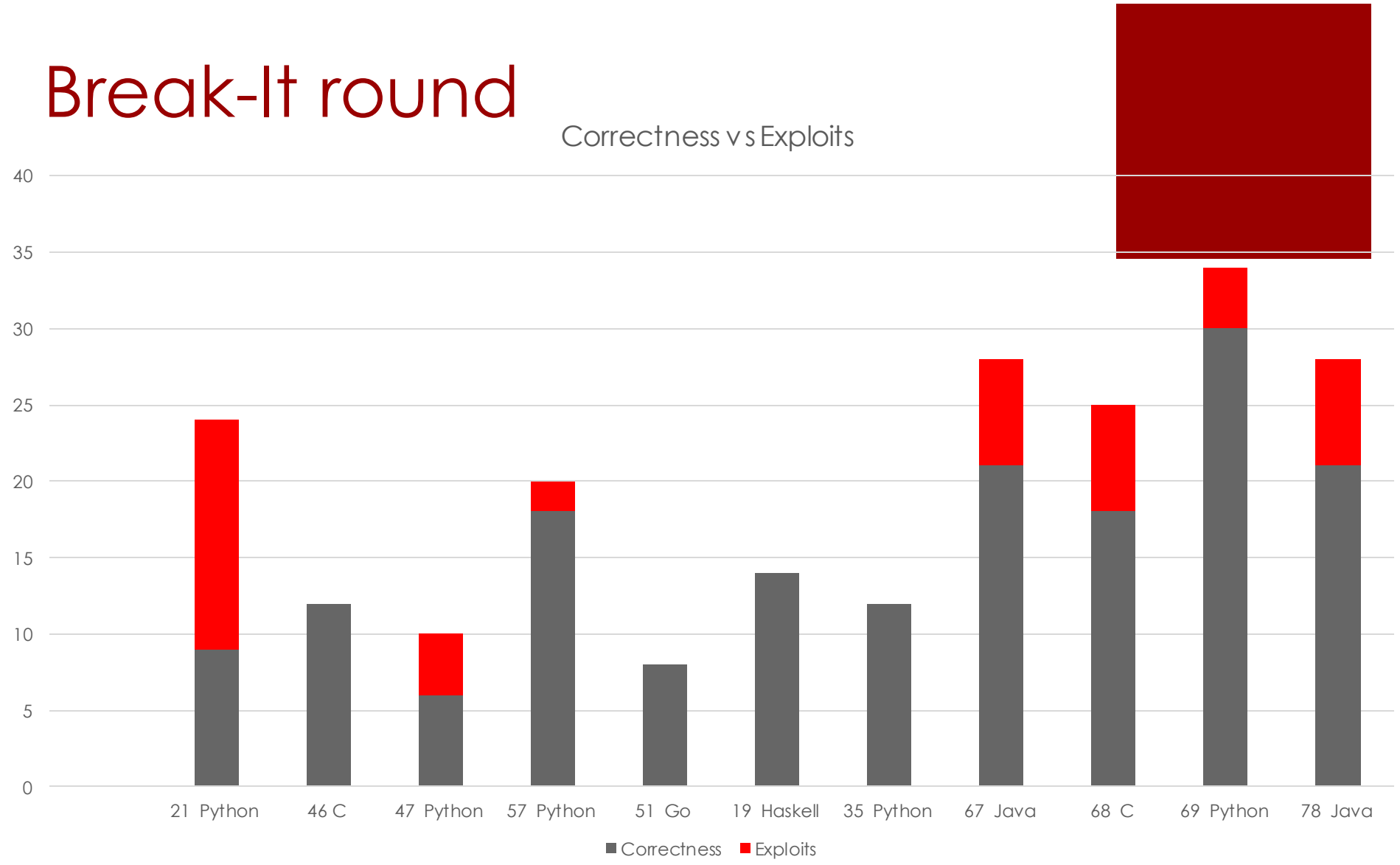


- We ran the contest over September
- Out of 90 registered **teams** with over 180 registered **individuals**, we had
  - **20** teams attempt to **submit** something
  - **11** teams submit code that passed core tests
- Successful submissions in Go, Haskell, Python, Java, C, and C++

Scores over time



Break-It round



Overall Winners

- First place build-it languages
  - Python
  - Haskell
- First place break-it team wrote in Go (and was third in build-it)

Some Fun Bugs

- Command line parsing leads to heap overflows
- Serialize data structures with `pickle.dumps`, then introduce replay attacks
- Serialize data structures with `Serializable`

What do we think about it?

- Memory safety helped but was not sufficient
  - This is an important property for the competition
- Strong static typing helped but was not enough
  - Python still wound up beating Haskell and Go

Security as state depth

- Consider the set of all states reachable in a program
- Hypothesis: A program has fewer bugs the smaller this state space is
- Can language features not related to memory safety improve the security of programs? Can we quantify this?

Stuff That Didn't Work

- Automatic correctness judging using "the wisdom of the crowds"
- My appendix
- Labor Day weekend
- Lots of small correctness bugs

Stuff That Did Work

- Infrastructure
  - A mixture of Haskell, Python and Java works shockingly well
- Security bug judging via confidentiality / integrity
- IRC

Future Contests

- Planning for Spring 2015
- New problem ideas
  - Remote command and control?
  - Voting?
- Increase scale, better automatic judging
- Only judge security bugs

How can you help?

- Sponsor the contest for prize money
- Contribute time as a professional breaker
- Contribute time as a judge
- Participate as a contestant

Conclusion

- Our contestants had fun and learned about security
- We measured peoples ability to both find bugs and write code
- We amplified one CTF problem into eleven
- We'll do it again

Acknowledgement

- UMD PI and staff
  - Dr. Michael Hicks, PI
  - Dr. Dave Levin, PI
  - Dr. Atif Memon, PI
  - Dr. Jan Plane, PI
  - James Parker
  - Kris Micinsiki
  - Nick Labich
  - Luís Pina

Acknowledgement

- Volunteer Judges
  - Stephen Chong (Harvard University)
  - John Regehr (University of Utah)
  - Stephen Magill, Ian Blumenfeld (Galois)
  - Aslan Askarov (Aarhus University)
  - Jeff Kuhn (Amches)
  - Lok Yan (AFRL)
  - Vu Le (University of California, Davis)
  - Alvaro Ugaz,Jay Ruhnke (SuperTEK)
  - Andre Protas (CyberPoint)
  - Bryon Fryer
  - Carl Steinebach
  - Austin Parker
  - Aaron Temin, Sue Wang (Mitre)

Thanks

- Support
  - National Science Foundation
- Corporate Sponsors
  - Cyberpoint
  - Trail of Bits
  - AT&T
  - SuprTEK
  - NCC Group
  - Cigital
  - Galois
  - Patriot
  - Astech Consulting
  - Amazon