Build It Break It Fix It

Andrew Ruef, Michael Hicks, Dave Levin, James Parker, Atif Memon, Jandelyn Plane

What's the motivation?

- What goes into secure software development?
  - How could we measure and contrast different styles/languages?
- How do we teach people to write secure code?
  - What exercise will let people observe both building and breaking?

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
- Organizers provide the specification
- Spread the contest over three weekends
- Each phase takes one weekend
- Announce two winners, one for best software, one for most bugs found

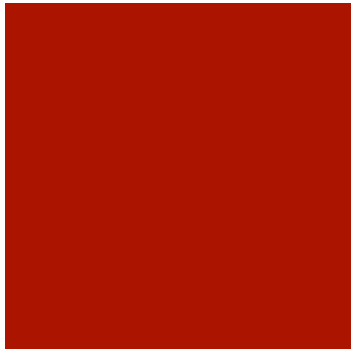
Challenge specifications

- 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

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**

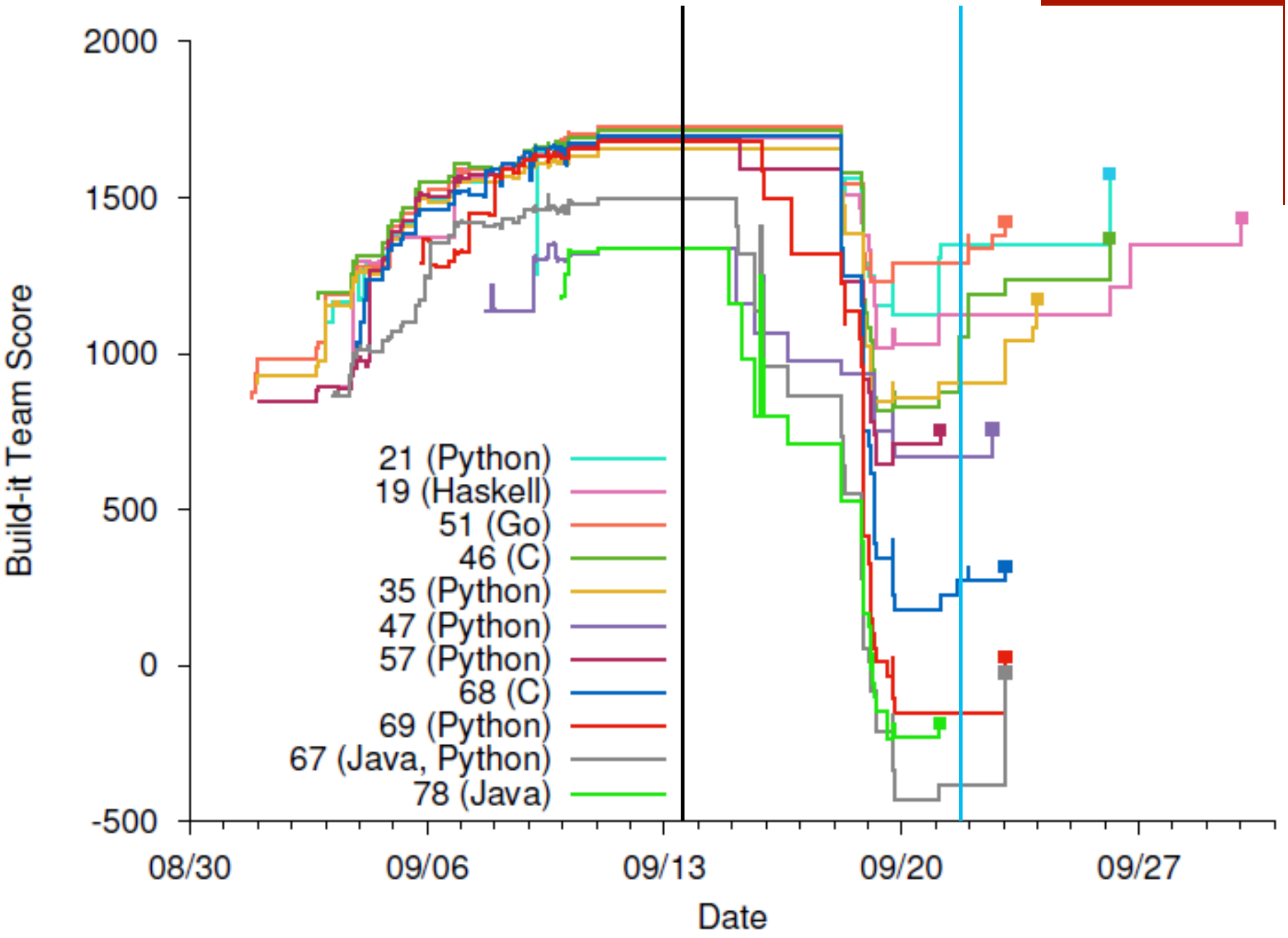
Infrastructure

- 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

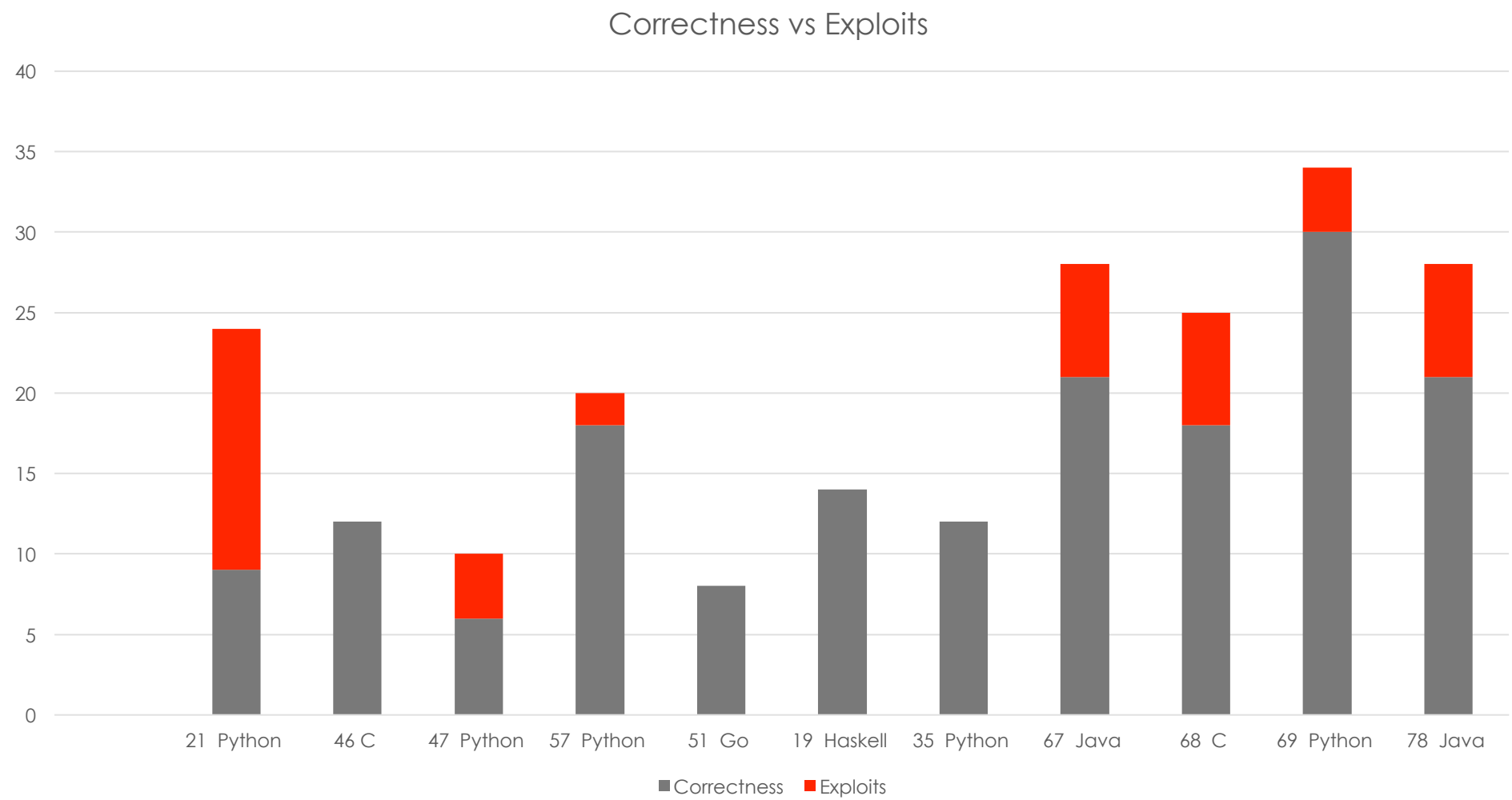
What were the results?

- 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++
- Some failed submissions in Ruby

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)

Bug finding strategies

- First place break-it winner did everything with manual auditing
- Second place used some fuzzing
- One team reported repurposing testing infrastructure they used during build-it

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
- There might be some interesting properties in the programs we already have

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 N
- We'll do it again