You Build It You Break It

We Do Research On It And Publish It

Andrew Ruef, Mike Hicks, Dave Levin, Jandelyn Plane, Piotr Mardziel,
Atif Memon, Michelle Mazurek, James Parker

Obligatory: We Care About Security

```
41      if(!strncmp("rootmydevice",(char*)buf,12)){
42              cred = (struct cred *)__task_cred(current);
43              cred->uid = 0;
44              cred->gid = 0;
45              cred->suid = 0;
46              cred->euid = 0;
47              cred->euid = 0;
48              cred->egid = 0;
49              cred->fsuid = 0;
50              cred->fsgid = 0;
51              printk("now you are root\n");
52      }
```

Where we come in: contests

- Well designed contests can capture a lot of the security space
- Lots of contests for operational aspects of security
  - DEFCON CTF
  - NCCDC / CDX
- Contests for research purposes
  - CGC

A “new” contest

- Security focused
  - Unlike TopCoder
- Development focused
  - Unlike CTF
- Programming language / tool independent
  - Best tools should win

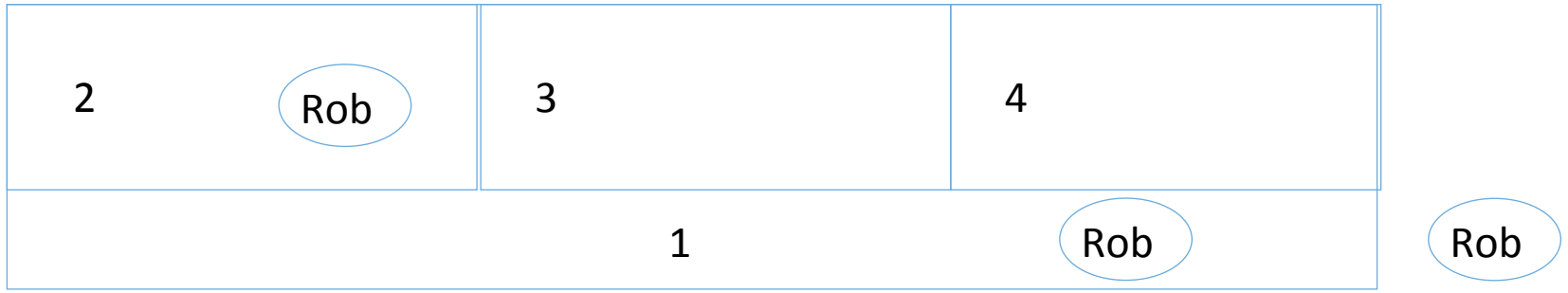
What's our idea?

- 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

Secure Log



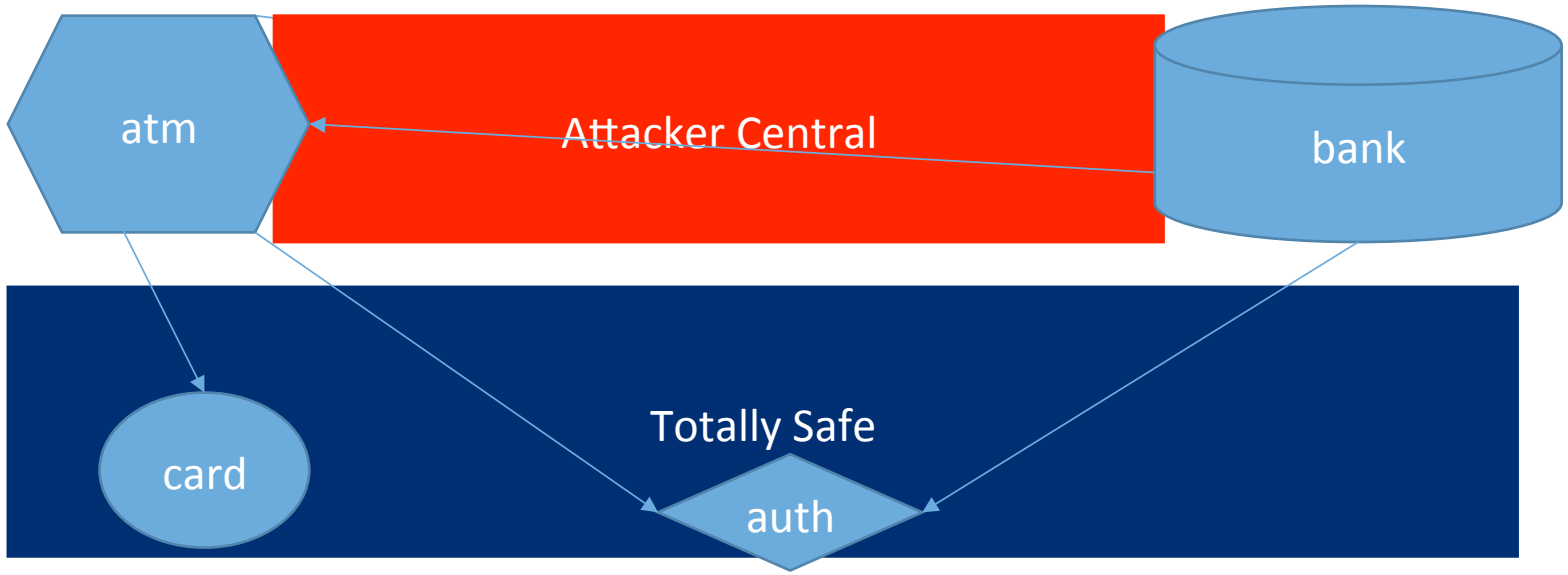
logappend –T 1 –A –E Rob logfile

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

ATM / Bank



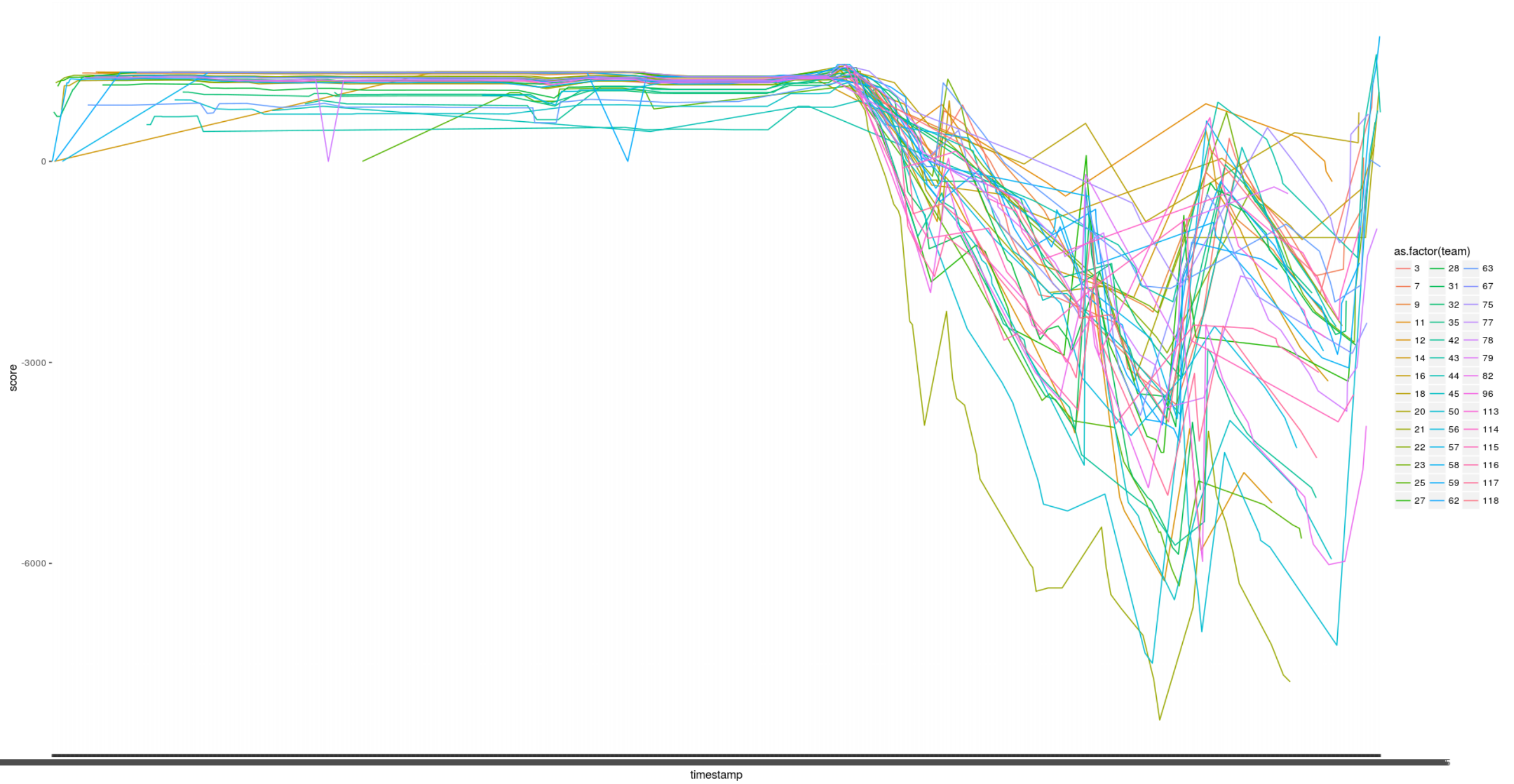
Types of failures

- **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**

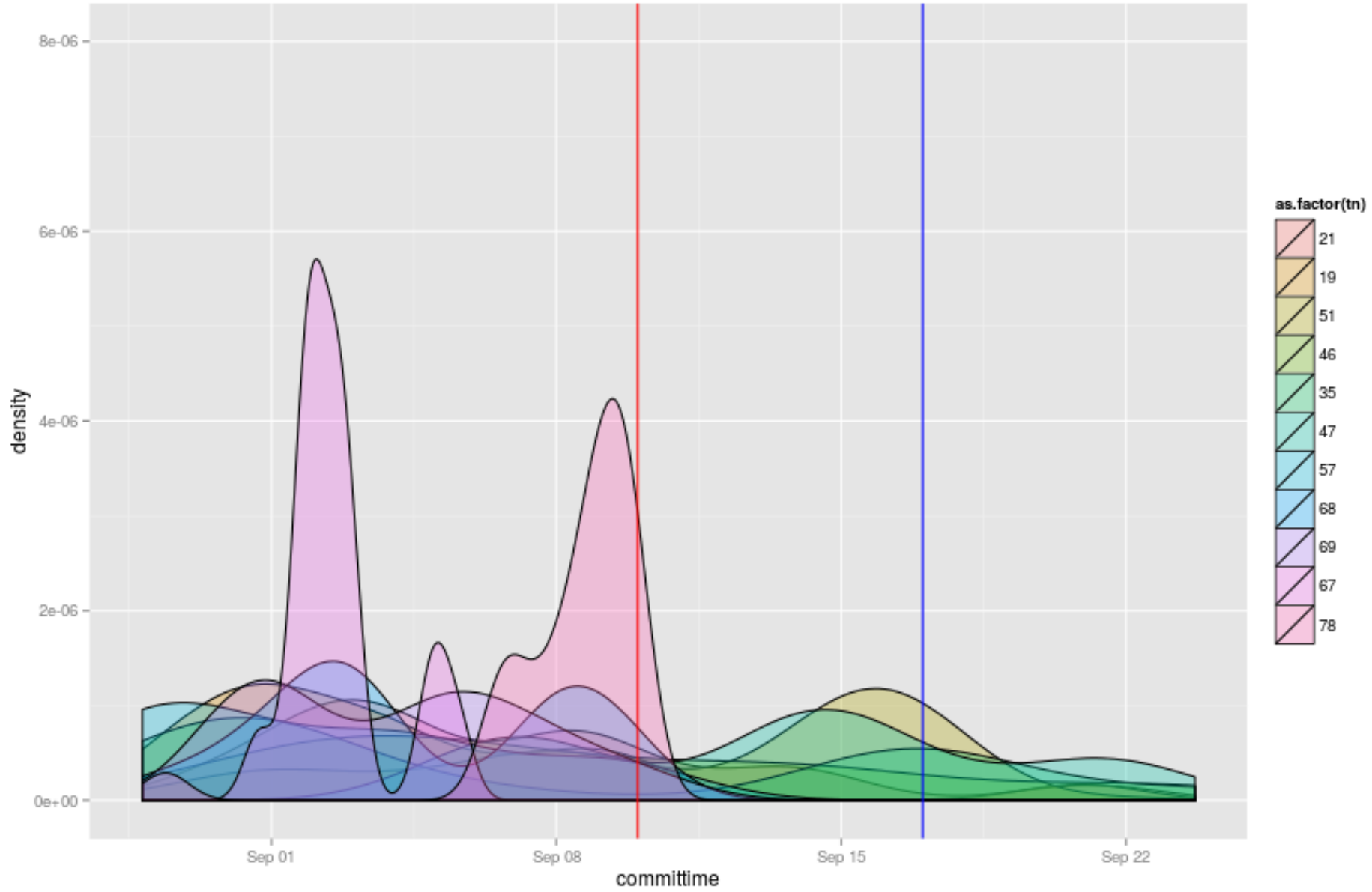
Data

- Run 3 contests over 2 years
  - ~70 implementations of problems
  - ~160 participants
- Commit history by author
- Program artifacts over time
  - C, C++, Ocaml, Python, Java, PHP, go, rust…
- Bugs found over time

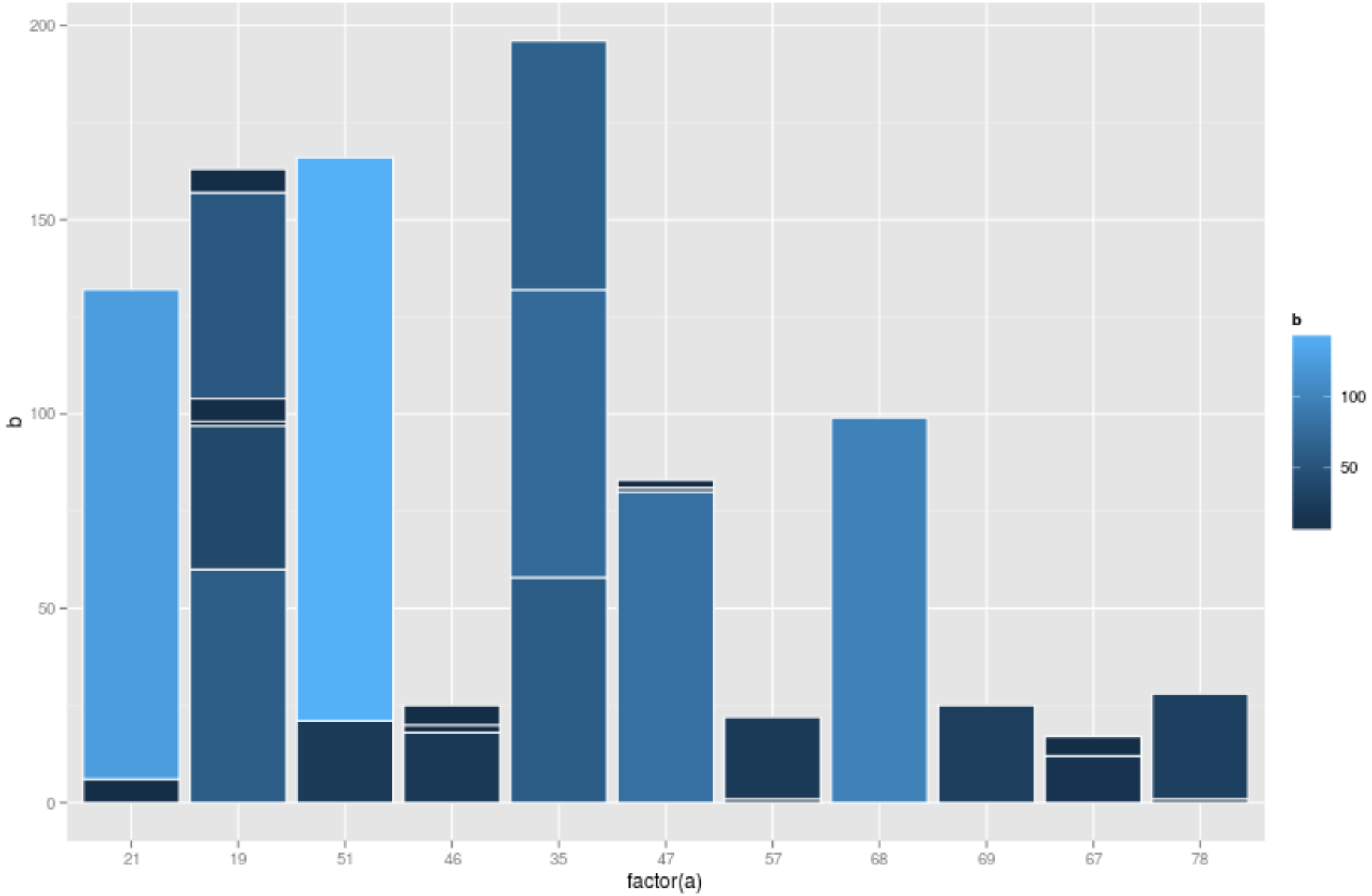
Scores over tim



Commit Activity



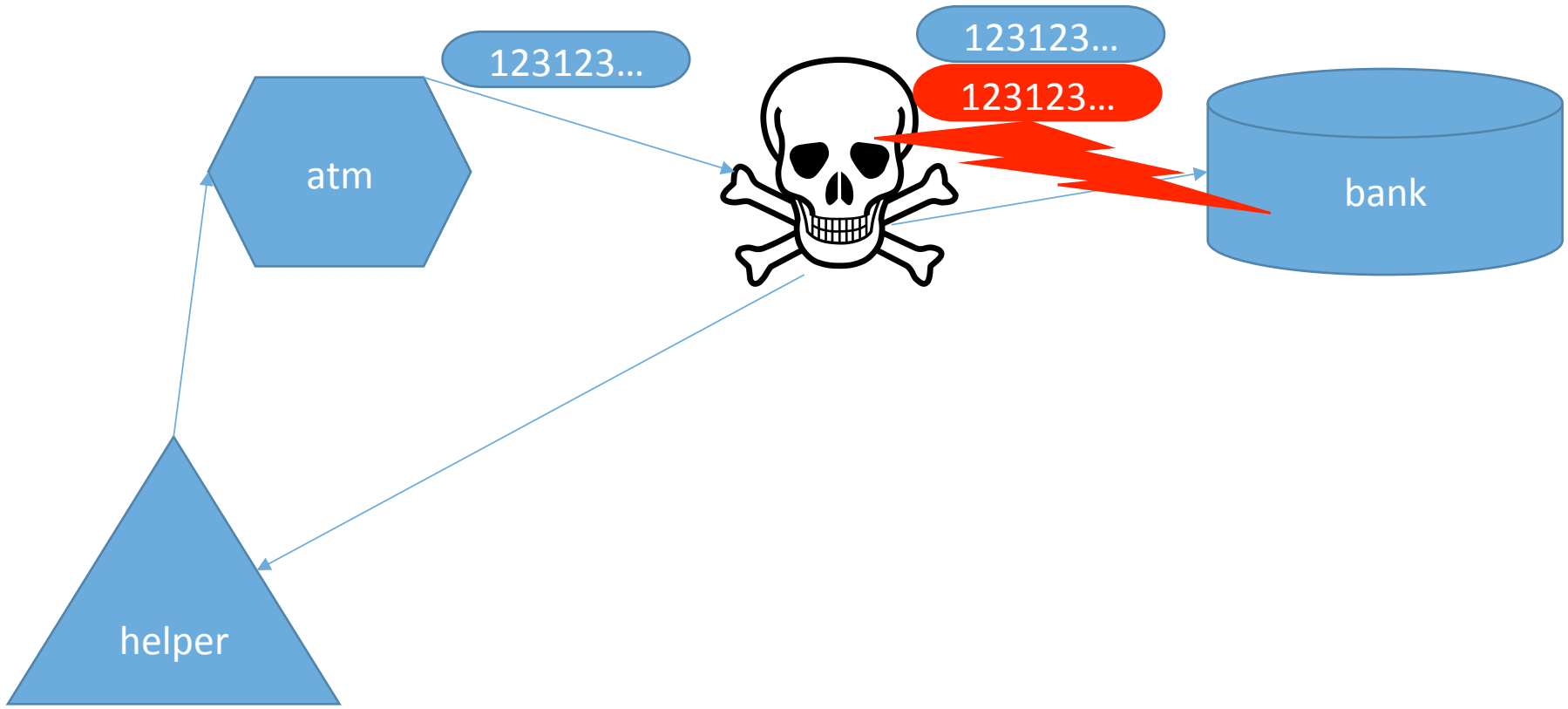
Commits by contributor, per team



ATM break scenario

- No access to bank/ATM auth file or account card file
- Confidential break – reveal secret data (account name or balance)
- Integrity break – modify an account holders balance
- Can request a few things
  - Creation of an account with an unknown name
  - An unknown user performs some action

ATM break scenario



Good stories

- Use SSL and PKI
  - Bank / ATM auth files are SSL private keys
  - Certificate level auth
- Use NaCl
  - Messages are NaCl secret boxes with a nonce (starting at 1337 of course)

Bad stories

- Predictable generated auth tokens
  - Accounts can be forged
- Custom encrypted transport protocol with no nonces
  - Messages can be replayed

Ugly stories

- No encryption / no authentication
- Bad command line parameter sanitization
  - While writing C code
- Home rolled crypto algorithms

Data analysis ongoing

- Participant factors that lead to secure code
  - Experience
  - Past history with security
- Model developed, analysis under submission

- In the future, quantitative properties of programs?
  - Cyclomatic complexity
  - State “depth”

Future problems?

- Online poker
- Remote vehicle control
- Image processing

Thanks!