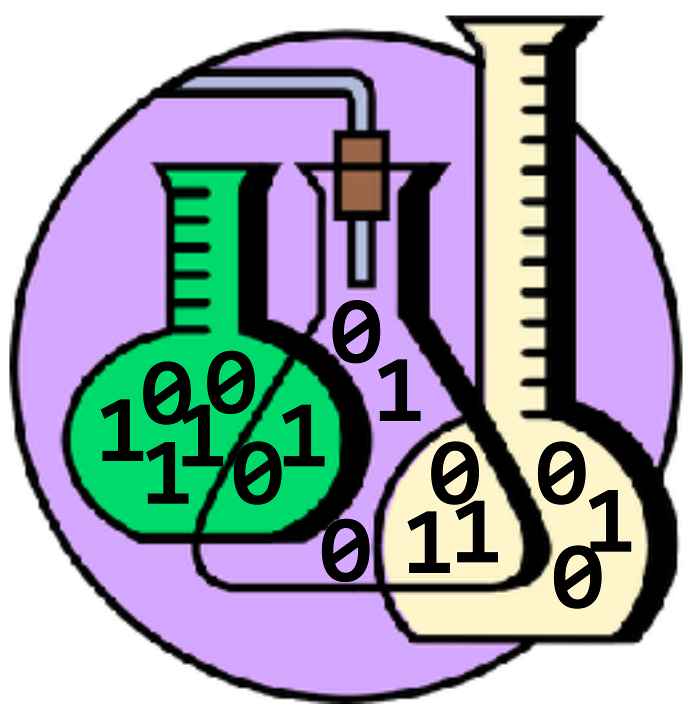
NYU TANDON SCHOOL OF ENGINEERING

Prospects and Pitfalls for a Science of Binary Analysis



Brendan Dolan-Gavitt

Binary Analysis Research

NYU

- Since the Cyber Grand Challenge, binary analysis has undergone something of a renaissance
- Lots of new (open!) tools, techniques
- Increased academic attention to long-ignored areas
  - Fuzzing – lots of work on why things like **AFL** work so well, and how to make them better
  - Measurements of how effective basic binary analyses (e.g., plain disassembly, function recognition) are
- New areas – function *similarity*, cross-architecture code search

How Good Are We?

- Before we pat ourselves on the back for a job well done and head off to the bar...
- How well are we doing in these areas?
- What are we still bad at – where should our research efforts be directed?

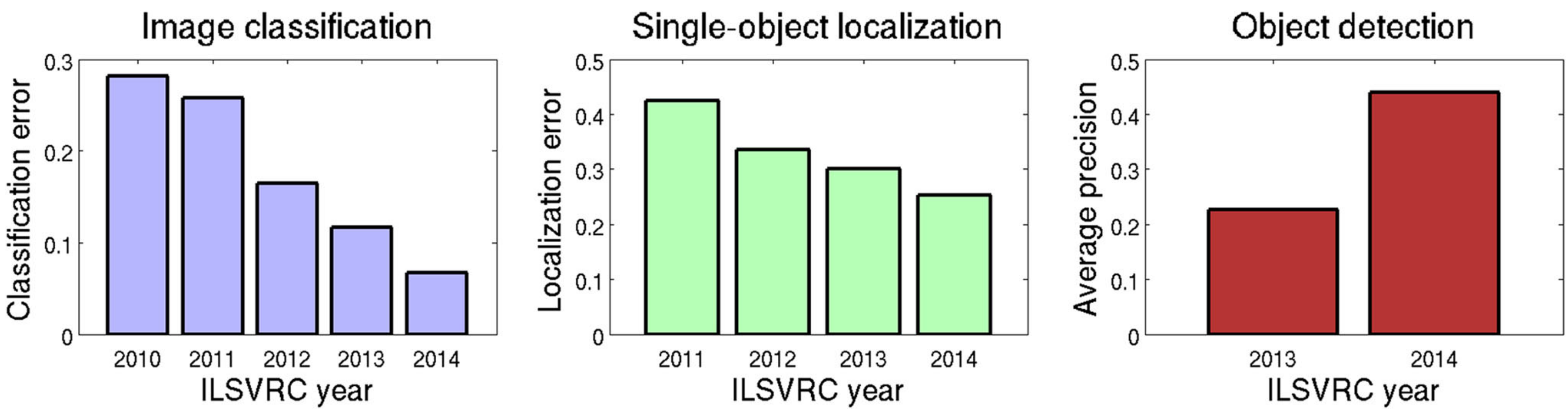
The Impact of Datasets

- The fastest way to make progress is through *open, well-labeled datasets*
  - Provide an easy source of test data for new algorithms
  - Standardization allows different approaches to be *compared*
  - Progress can be measured over time!

Case Study: IMAGENET

- ImageNet introduced by Fei-Fei Li's group in 2009
- 14 *million* images, annotated with labels from WordNet
- Annual image recognition competition: ILSVRC (2010-2017)
  - Competition made it clear how much progress the field was making
  - Helped catalyze **huge** improvements in image recognition algorithms:

ImageNet Progress



Source:

ImageNet Large Scale Visual Recognition Challenge

Olga Russakovsky[1] · Jia Deng[2] · Hao Su[1] · Jonathan Krause[1] · Sanjeev Satheesh[1] · Sean Ma[1] · Zhiheng Huang[1] · Andrej Karpathy[1] · Aditya Khosla[3] · Michael Bernstein[1] · Alexander C. Berg[4] · Li Fei-Fei[1]

2017 Update:
Classification error **0.02**

This Talk

- What datasets do we have in binary analysis?
- What do we need, and what are the pitfalls?
- Walk through public datasets in three key areas:
  - Bugs and vulnerabilities
  - Dynamic malware analysis
  - Function recognition in binaries

NYU

Vulnerability Discovery

- Finding vulnerabilities in software automatically has been a major research and industry goal for the last 25 years

Academic

An Empirical Study of the Reliability of UNIX Utilities

Barton P. Miller
bart@cs.wisc.edu

Lars Fredriksen
L.Fredriksen@att.com

Bryan So
so@cs.wisc.edu

Fuzzing (1989)

KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs

Cristian Cadar, Daniel Dunbar, Dawson Engler *
Stanford University

KLEE (2005)

Driller: Augmenting Fuzzing Through Selective Symbolic Execution

Nick Stephens, John Grosen, Christopher Salls, Andrew Dutcher, Ruoyu Wang, Jacopo Corbetta, Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna
UC Santa Barbara
{stephens,jmg,salls,dutcher,fish,jacopo,yans,chris,vigna}@cs.ucsb.edu

Driller (2015)

Commercial

klocwork — a Rogue Wave Company

∀.Secure

FORTIFY

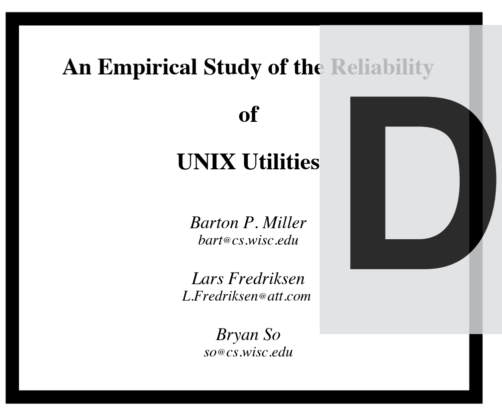
VERACODE

coverity

Vulnerability Discovery

- Finding vulnerabilities in software automatically has been a major research and industry goal for the last 25 years

Academic

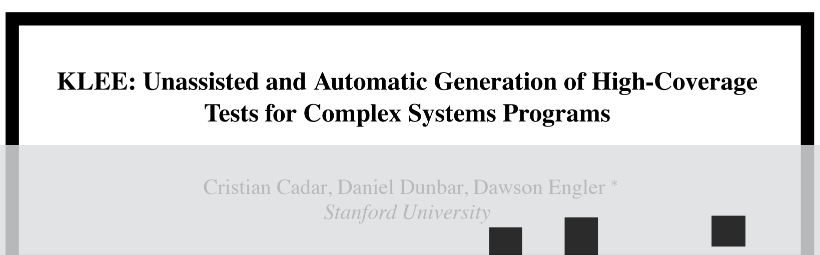

An Empirical Study of the Reliability
of
UNIX Utilities

Barton P. Miller
bart@cs.wisc.edu

Lars Fredriksen
L.Fredriksen@att.com

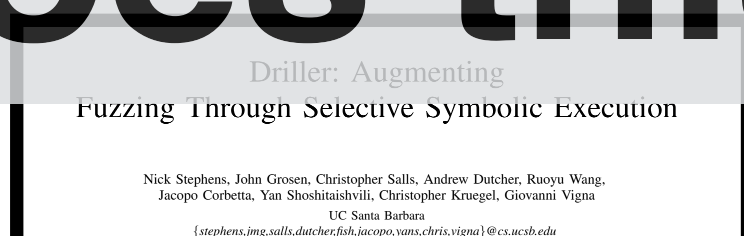
Bryan So
so@cs.wisc.edu

Fuzzing (1989)


KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs

Cristian Cadar, Daniel Dunbar, Dawson Engler *
Stanford University

KLEE (2005)


Driller: Augmenting Fuzzing Through Selective Symbolic Execution

Nick Stephens, John Grosen, Christopher Salls, Andrew Dutcher, Ruoyu Wang, Jacopo Corbetta, Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna
UC Santa Barbara
{stephens,jmg,salls,dutcher,fish,jacopo,yans,chris,vigna}@cs.ucsb.edu

Driller (2015)

Commercial

klocwork
a Rogue Wave Company
∀.Secure
FORTIFY
VERACODE
coverity®

Does this work??

Bug and Vulnerability Corpora

- NIST's SAMATE project collects data sets and runs an annual "bake-off" – but competitors are not named
- Their Software Assurance Reference Dataset (SARD) contains many sub-datasets
  - Juliet: C/C++ and Java programs with bugs
  - IARPA STONESOUP: injected bugs
  - Toyota InfoTechnology Center static analysis benchmarks

NYU

New Kids on the Block

LAVA

CYBER
GRAND_CHALLENGE

DARPA Cyber Grand Challenge

- Held by DARPA in 2015 (CQE) and 2016 (CFE)
- Fully automated hacking machines – cool!
- Even cooler: dataset of 247 reasonably-sized C and C++ programs
  - Variety of vulnerabilities
  - Interaction required
  - Each comes with normal and triggering inputs

CGC Ports

- Trail of Bits has ported a large number of the CGC challenges to Linux
- This (in theory) lets off-the-shelf tools be evaluated
- In practice, still many barriers
  - (Ask me about my attempts to get KLEE running on the CGC dataset sometime...)
- They would **love** help finishing the porting effort!
- Available: https://github.com/trailofbits/cb-multios

Downsides of CGC

- Static dataset: there was one CGC, and that's all we get
- Some artificial features:
  - 7 system calls total - doesn't reflect complexity of real OS environments
  - Single architecture (32-bit x86)
  - Random "flag" page at fixed offset

LAVA Corpora

- In 2016, we[1] created a *bug injection* system that can add thousands of bugs (mem corruption) to existing programs
- Each bug comes with a triggering input
- Bugs are synthetic but (we hope) good proxies for real bugs (at least for automated tools)

1. We =

Available Datasets

- **LAVA-M**: 4 coreutils programs (md5sum, uniq, base64, and who) with bugs injected
- **LAVA-1**: 69 versions of file, each with one bug
- "**Toy**" dataset: 159 versions of a 70 line C program
  - Useful for finding bugs in bug finders!
  - By default, KLEE only finds the bug in 43%
- http://moyix.blogspot.com/2016/10/the-lava-synthetic-bug-corpora.html

Progress on LAVA-M

- Several papers have used LAVA-M to evaluate new fuzzers

Program	# Bugs	Vuzzer	Steelix	SBF
base64	44	17	43	**44**
md5sum	57	1	**28**	–
uniq	28	**27**	7	–
who	2136	50	**194**	–

- We can see that the original LAVA-M programs are almost "used up" – time to create new corpora!

Malware Analysis

- One of the core use cases for binary analysis is automated analysis of *malicious* software
- Some public corpora exist for *static* malware features
  - Microsoft has a dataset on Kaggle with 400GB of samples from 9 families – headers stripped
- But no similar dataset available for *dynamic* analysis

MalRec: A Full-Trace Malware Corpus for Dynamic Analysis

- Based on PANDA – dynamic analysis platform we developed
- By using *non-deterministic record and replay*, we can capture all malware behavior – down to individual instructions
- Currently processes **100 malware samples** per day; has been running for **3 years**
- Because record/replay captures *all* information, we can **retroactively** capture features of interest

Malrec Stats

- More than 100,000 traces available for download
- More than 1.5 quadrillion instructions' worth of execution
- Because of record/replay and some compression tricks, this dataset is only 3.5 TB
- Available:
  http://panda.moyix.net/~moyix/rr/
  http://giantpanda.gtisc.gatech.edu/malrec/rr/README

Malrec Shortcomings

- No attempt to mask emulator features, so lots of evasion: at least 10% (conservative estimate)
- Unclear if sample is representative of all malware!
- As with all malware datasets, no ground truth labels
    - But we hope that since these are full traces we can improve ground truth over time

Function Identification

- Andriesse et al. (USENIX Sec 2016), noted that although *disassembly* is now very reliable, *function identification* is not
  - Up to 20% false negative rates for funtion starts with IDA Pro
  - Some false positives too

ByteWeight Dataset

- Binaries from open-source programs: coreutils, binutils, findutils on Linux, putty, 7zip, vim, libsodium, libetpan, HID API, and pbc on Windows
- Three compilers (gcc/clang/icc), four optimization levels, two operating systems, both 32- and 64-bit x86
- Used for evaluating ByteWeight (Bao et al., 2014) and a later neural network-based approach by Shin et al. (2015)
- Available: http://security.ece.cmu.edu/byteweight/

ByteWeight **Warning**

- Subtle gotcha (Andriesse et al., 2017): coreutils programs *share large amounts of library code* – average coreutils binary shares 94% of its functions with at least one other binary!

- This means that for machine learning purposes, standard training set / test set split will have many overlaps

- This can lead to misleading results when machine learning-based techniques are used – you're testing on your training data!

- (This is not a knock on Bao et al. – if their data weren't open & available, would have been hard to spot this!)

Vector35 Dataset

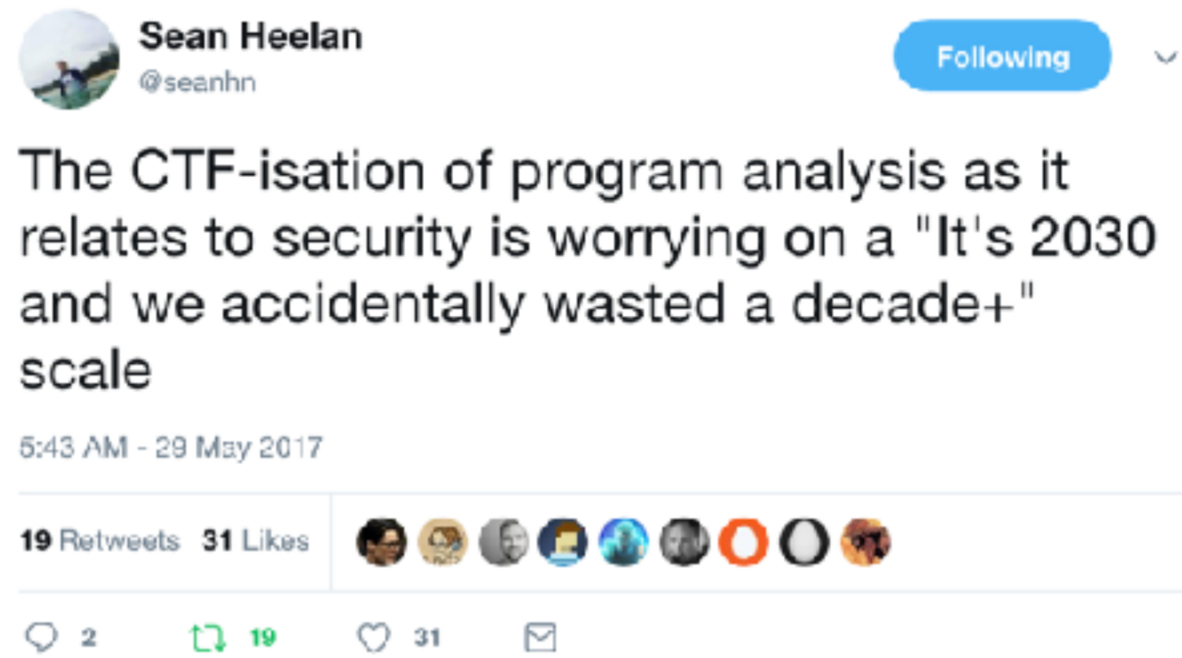
- Recently, Vector35 (creators of Binary Ninja) put together a *cross-architecture* dataset used for testing their own tools
- Combination of:
    - Original ByteWeight dataset
    - DARPA CGC binaries (clang, 32-bit)
    - Busybox (six architectures, gcc, two levels of optimization)

Dataset Pitfalls

- Although I believe standard datasets are on the whole a huge win for research, there are some dangers too
- The most pressing concern is **validity** – datasets are inherently *approximations* of our real problem
- When our tools do well on our datasets, do they translate to the real world?

NYU

Validity



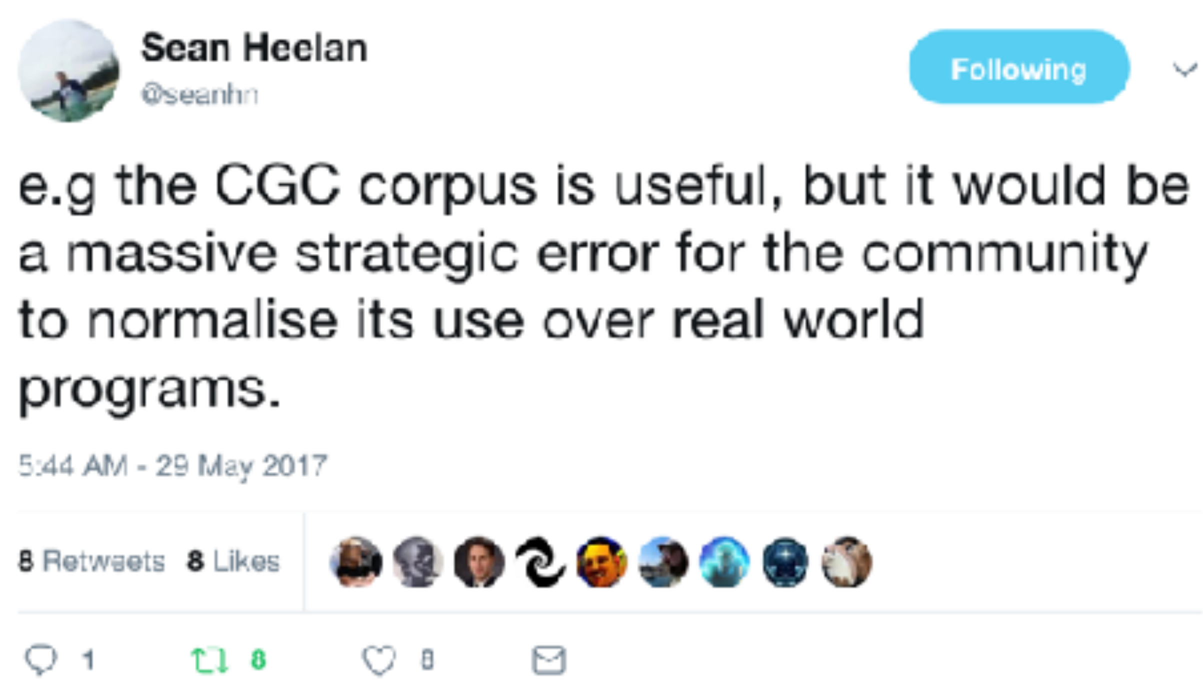
Sean Heelan
@seanhn
Following

The CTF-isation of program analysis as it relates to security is worrying on a "It's 2030 and we accidentally wasted a decade+" scale

5:43 AM - 29 May 2017

19 Retweets 31 Likes

2 19 31



Sean Heelan
@seanhn
Following

e.g the CGC corpus is useful, but it would be a massive strategic error for the community to normalise its use over real world programs.

5:44 AM - 29 May 2017

8 Retweets 8 Likes

1 8 8

No Easy Answers

- You can say "well just try it on real software too!"
- But you can't try *all* real software – and so any subset you pick may also be biased!
- Instead, we can try to measure dataset bias in a few indirect ways
  - (These come from "Unbiased Look at Dataset Bias" by Torralba and Efros)

Cross-Dataset Generalization

- One simple test is *cross-dataset generalization*
  - How well does my technique work when I try it on someone else's data?
- For this we need *more than one* large, public dataset!
- Right now, we often have just one for a task in binary program analysis

Negative Dataset Bias

- In many tasks, it is just as important to have good representation of **negative** examples as positive
  - ML example – to recognize boats, need lots of images of things that have water but are *not* boats
- For some datasets, this is relatively easy (binaries with function starts have lots of data that are *not* function starts)
- Harder in other cases – representative datasets of *benign* software?
- One way to evaluate negative dataset bias: train on positives & negatives from your own dataset, test on negatives from many datasets combined

Conclusions

- Large, well-labeled public datasets are **crucial** to achieving rapid progress in binary analysis
- We have made some recent progress as a community – let's keep it up!
  - Improve current datasets, and create new ones!
  - Check for bias and mistakes in existing datasets
- If you create a new analysis, you should:
  - Test it on an existing dataset if there is one
  - **Release** your dataset!

Rejected Titles

- Measure-y Loves Company
- Habeas Corpora
- We Gave Researchers Standardized Datasets – You Won't **Believe** What Happened Next!