CYBERSECURITY AT UNIVERSITY OF IDAHO

Jim Alves-Foss
University of Idaho
UNIV. OF IDAHO CYBERSECURITY

Education:
- We offer undergraduate and graduate Certificates in cybersecurity.
- We offer a BS and MS in Cyber security. The BS consists of several CS courses and 12 Cybersecurity specific courses. The MS requires four cybersecurity specific courses, and several electives with thesis or project.
- Hands-on air-gapped lab. Power systems lab.

Research
- Binary analysis, design and analysis of secure systems, industrial controls system security (specifically with respect to power grid), forensics, side-channel analysis, secure software engineering

Staff
- 12+ faculty: 6 CS, 5 ECE, 1+ Sociology, 1+ Business

History
- Offered first cybersecurity course in 1992.
- 6 CS cybersecurity faculty, 3 ECE cyber security faculty, plus others with interest.
PERCENTAGES, PROBABILITIES AND PROFESSIONS OF PERFORMANCE

Jim Alves-Foss
University of Idaho
Several papers in computer science/cybersecurity compare the results of some tool or algorithm to:

- Other tools/algorithms
- Across different datasets

The papers are written to show successful results, but often with limited information and poor statistics.
EXAMPLE: THE PROBLEM

Compilers translate source code into binary images

- Stripped binaries do not have symbol tables
- It is not obvious where functions start and end
- Problem exacerbated by
  - Different compilers
  - Different compiler options
  - Programming language choices
BENEFITS OF A SOLUTION

Reverse Engineering and Decompilation

Automated Analysis
- Unit level testing of binaries

Binary Patching
- Correcting security flaws (Goal of DARPA CGC and AMP programs)
- Inserting security protections
  - control flow integrity, stack guards
PRIOR ACADEMIC ATTEMPTS

Academic

- Bao et al developed a machine-learning based approach
  - tested on Unix utilities for gcc and icc
  - appears that results incorporated into BAP
- Shin et al developed a neural network approach
  - tested on Unix utilities for gcc and icc
  - did not follow through on this
- Andriesse et al developed a graph-theoretic approach (*Nucleus*)
  - tested on Unix utilities, SPEC CPU 2006 and some servers for gcc and clang
PRIOR COMMERCIAL ATTEMPTS

Commercial

- IDA Pro automated process (tested with free version)
- NSA’s Ghidra
HOW GOOD ARE PRIOR SOLUTIONS?

Following model of prior work used 3 datasets

- Unix utilities: binutils, coreutils, findutils (129 unique programs)
- SPEC CPU 2017 (INTSPEED, INTRATE, FPSPEED, FPRATE)
  - Includes some Fortran code
  - Includes C/C++ with some exception handling
  - 28 unique 32-bit programs and 32 unique 64-bit programs
  - Some require –fpic (position independent code) compilation
- All compiled with gcc (gfortran), clang and icc (ifort) compilers
  - using –O0 through –O3 optimization flags

- Chrome Browser: compiled with default (clang and –O2)
HOW DO WE MEASURE SUCCESS?

How do we measure success?

- We use symbol table from unstripped version of test suites for “ground truth”
- We measure:
  - True Positive (TP), False Positive (FP), False Negative (FN)
  - Function Starts:
    - Did we get the right location?
    - What if symbol tables has aliases?
  - Function Boundaries:
    - Did we get the correct length?
    - Is short ok?
    - What if symbol table shows 0 length?
    - Does symbol table length include padding bytes?
HOW DO WE MEASURE SUCCESS (2)?

Prior work uses the following:

- **Precision (P):** How correct are our answers: \( \frac{TP}{TP + FP} \)
  - High precision means low false positive rate
- **Recall (R):** How many functions did we find: \( \frac{TP}{TP + FN} \)
  - High recall means low false negative rate
- **F1:** weighted average of P and R: \( \frac{2 \times P \times R}{P + R} \)

Results are based on the means of results from datasets.
WHAT DOES MEAN MEAN?

You use the same dataset for all tools. Your numbers are bigger/better than their numbers.

Are you always better?

- If not, where and why?

Are there a few instances where you are a lot better and these skew the means?
MEANS OF PERCENTAGES?

Percentages miss the big picture – what does it really mean to have a higher percentage?

- Are you better for all test cases?
- Are you a lot better in just some tests cases?
### EXAMPLES PERCENTAGES...

Table 1: Precision Recall of function start identification (reproduction of Table 2 from Bao et al. [3])

<table>
<thead>
<tr>
<th></th>
<th>GCC</th>
<th></th>
<th>ICC</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>Time (sec)</td>
<td>Precision</td>
</tr>
<tr>
<td>Rosenblum et al. [7]</td>
<td>0.4909</td>
<td>0.4312</td>
<td>1172.41</td>
<td>0.6080</td>
</tr>
<tr>
<td>BYTEWEIGHT (3)</td>
<td>0.9103</td>
<td>0.8711</td>
<td>1417.51</td>
<td>0.8948</td>
</tr>
<tr>
<td>BYTEWEIGHT (no-norm)</td>
<td>0.9877</td>
<td>0.9302</td>
<td>19994.18</td>
<td>0.9727</td>
</tr>
<tr>
<td>BYTEWEIGHT</td>
<td>0.9726</td>
<td>0.9599</td>
<td>1468.75</td>
<td>0.9725</td>
</tr>
</tbody>
</table>

Table 2: Precision Recall of function start identification (reproduction of Table 1(a) from Andriesse et al. [1])

<table>
<thead>
<tr>
<th></th>
<th>gcc x86</th>
<th>gcc x64</th>
<th>clang x86</th>
<th>clang x64</th>
<th>VS x86</th>
<th>VS x64</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDA Pro 6.7</td>
<td>0.98/0.78</td>
<td>0.97/0.74</td>
<td>0.98/0.78</td>
<td>0.98/0.77</td>
<td>0.84/0.93</td>
<td>1.00/0.94</td>
</tr>
<tr>
<td>BAP/ByteWeight 0.9.9</td>
<td>0.68/0.83</td>
<td>0.70/0.66</td>
<td>0.52/0.71</td>
<td>0.73/0.49</td>
<td>0.63/0.74</td>
<td>0.69/0.56</td>
</tr>
<tr>
<td>Dyninst 9.1.0</td>
<td>0.93/0.91</td>
<td>0.96/0.74</td>
<td>0.98/0.95</td>
<td>0.88/0.72</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Nucleus</td>
<td>0.98/0.96</td>
<td>0.98/0.96</td>
<td>0.96/0.97</td>
<td>0.96/0.95</td>
<td>0.86/0.96</td>
<td>0.95/0.94</td>
</tr>
<tr>
<td>△Nucleus</td>
<td>+0.00/+0.05</td>
<td>+0.01/+0.22</td>
<td>–0.02/+0.02</td>
<td>–0.02/+0.18</td>
<td>+0.02/+0.03</td>
<td>–0.05/+0.00</td>
</tr>
</tbody>
</table>
WHAT IS THE NATURE OF YOUR RESULTS

An average is just a single value. You can do better than that.

Visualize the Data
HISTOGRAM OF F1 VALUES FOR COMBINED DATASETS
SCATTERPLOT OF F1 VALUES FOR COMBINED DATASETS (SORTED BY TEST NAME)

Interesting
SCATTERPLOT FOR UNIX UTILITIES DATASET,
(GROUPED BY OPTIMIZATION LEVELS AND
SORTED BY NAME)

Still Interesting
SCATTERPLOT FOR UNIX UTILITIES DATASET, (GROUPED BY OPTIMIZATION LEVELS AND SORTED BY FILESIZE)
Now, let’s look at the following visualizations when comparing tools for function boundary (specifically function start) identification. *(Recall the tables)*

### Table 1: Precision Recall of function start identification (reproduction of Table 2 from Bao et al. [3])

<table>
<thead>
<tr>
<th></th>
<th>GCC</th>
<th></th>
<th></th>
<th>ICC</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>Time(sec)</td>
<td>Precision</td>
<td>Recall</td>
<td>Time(sec)</td>
</tr>
<tr>
<td>Rosenblum et al. [7]</td>
<td>0.4909</td>
<td>0.4312</td>
<td>1172.41</td>
<td>0.6080</td>
<td>0.6749</td>
<td>2178.14</td>
</tr>
<tr>
<td>BYTEWEIGHT (3)</td>
<td>0.9103</td>
<td>0.8711</td>
<td>1417.51</td>
<td>0.8948</td>
<td>0.8592</td>
<td>1905.34</td>
</tr>
<tr>
<td>BYTEWEIGHT (no-norm)</td>
<td>0.9877</td>
<td>0.9302</td>
<td>19994.18</td>
<td>0.9727</td>
<td>0.9132</td>
<td>20894.45</td>
</tr>
<tr>
<td>BYTEWEIGHT</td>
<td>0.9726</td>
<td>0.9599</td>
<td>1468.75</td>
<td>0.9725</td>
<td>0.9800</td>
<td>1927.90</td>
</tr>
</tbody>
</table>

### Table 2: Precision Recall of function start identification (reproduction of Table1(a) from Andriese et al. [1])

<table>
<thead>
<tr>
<th></th>
<th>gcc x86</th>
<th>gcc x64</th>
<th>clang x86</th>
<th>clang x64</th>
<th>VS x86</th>
<th>VS x64</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDA Pro 6.7</td>
<td>0.98/0.78</td>
<td>0.97/0.74</td>
<td>0.98/0.78</td>
<td>0.98/0.77</td>
<td>0.84/0.93</td>
<td>1.00/0.94</td>
</tr>
<tr>
<td>BAP/ByteWeight 0.9.9</td>
<td>0.68/0.83</td>
<td>0.70/0.66</td>
<td>0.52/0.71</td>
<td>0.73/0.49</td>
<td>0.63/0.74</td>
<td>0.69/0.56</td>
</tr>
<tr>
<td>Dyninst 9.1.0</td>
<td>0.93/0.91</td>
<td>0.96/0.74</td>
<td>0.98/0.95</td>
<td>0.88/0.72</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Nucleus</td>
<td>0.98/0.96</td>
<td>0.98/0.96</td>
<td>0.96/0.97</td>
<td>0.96/0.95</td>
<td>0.86/0.96</td>
<td>0.95/0.94</td>
</tr>
<tr>
<td>△Nucleus</td>
<td>+0.00/+0.05</td>
<td>+0.01/+0.22</td>
<td>-0.02/+0.02</td>
<td>-0.02/+0.18</td>
<td>+0.02/+0.03</td>
<td>-0.05/+0.00</td>
</tr>
</tbody>
</table>
BAP - Function Starts (F1 values)

- Clang (61.1%)
- GCC (93.2%)
- ICC (84.3%)
Nucleus - Function Starts (F1 values)

- Clang (95.2%)
- GCC (97.6%)
- ICC (73.7%)
Ghidra - Function Starts (F1 values)

- Clang (85.9%)
- GCC (99.3%)
- ICC (75.5%)
IDA PRO (FREE) - Function Starts (F1 values)

GCC (83.1%)
JIMA - Function Starts (F1 values) – as of Oct. 2019

Clang (99.9%)  GCC (100.0%)  ICC (98.9%)
HOW ACCURATE IS ACCURACY?

Confusion Matrix

<table>
<thead>
<tr>
<th>Condition Positive</th>
<th>Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicted Positive</strong></td>
<td><strong>True Positive</strong></td>
</tr>
<tr>
<td><strong>Predicted Negative</strong></td>
<td><strong>False Negative (Type 2 error)</strong></td>
</tr>
</tbody>
</table>
SOME DEFINITION FROM CONFUSION MATRIX

\[
\text{Prevalence} = \frac{\sum \text{Condition positive}}{\sum \text{Total Population}}
\]

\[
\text{Accuracy} = \frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total Population}}
\]

\[
\text{Precision} = \frac{\sum \text{True positive}}{\sum \text{Predicted positive}}
\]

\[
\text{Recall} = \frac{\sum \text{True positive}}{\sum \text{Condition positive}}
\]

\[
F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]
CONSIDER THE FOLLOWING **GROUND TRUTH** (WHERE THERE ARE RARE INSTANCES OF CONDITION WE ARE LOOKING FOR)

<table>
<thead>
<tr>
<th></th>
<th>Condition Positive</th>
<th>Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicted Positive</strong></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><strong>Predicted Negative</strong></td>
<td></td>
<td>990</td>
</tr>
</tbody>
</table>
99% ACCURATE

You found nothing, and are 99% accurate, but not very useful

<table>
<thead>
<tr>
<th></th>
<th>Condition Positive</th>
<th>Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>10</td>
<td>990</td>
</tr>
</tbody>
</table>
### Precision, Recall and F1 Value

<table>
<thead>
<tr>
<th></th>
<th>Condition Positive</th>
<th>Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>10</td>
<td>990</td>
</tr>
</tbody>
</table>

Precision is 0%, Recall is 0%, F1 is 0%
### PRECISION, RECALL AND F1 VALUE (NEW EXAMPLE)

<table>
<thead>
<tr>
<th></th>
<th>Condition Positive</th>
<th>Condition Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Predicted Negative</td>
<td>5</td>
<td>985</td>
</tr>
</tbody>
</table>

Precision is $5/10 = 50\%$, Recall is 50\%, F1 is 50\% (But I am still 99\% accurate!)

These areas are relevant
KNOW WHAT IS BEING MEASURED

- Understand what you are measuring
- Understand the statistics
- Present it reliably
KNOW WHAT IS BEING MEASURED (2)

zlib_zlib_uncompress_fuzzer summary

Ranking by median reached coverage

Reached coverage distribution

Fuzzer (highest median coverage on the left)

Mean coverage growth over time

* The error bands show the 95% confidence interval around the mean coverage.
Coverage Report

Created: 2020-09-07 05:48

Click [here](#) for information about interpreting this report.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Function Coverage</th>
<th>Line Coverage</th>
<th>Region Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>zlib/adler32.c</td>
<td>40.00% (2/5)</td>
<td>70.10% (68/97)</td>
<td>75.00% (105/140)</td>
</tr>
<tr>
<td>zlib/crc32.c</td>
<td>0.00% (0/13)</td>
<td>0.00% (0/333)</td>
<td>0.00% (0/76)</td>
</tr>
<tr>
<td>zlib/inffast.c</td>
<td>100.00% (1/1)</td>
<td>74.89% (164/219)</td>
<td>64.29% (54/84)</td>
</tr>
<tr>
<td>zlib/inflate.c</td>
<td>45.45% (10/22)</td>
<td>59.55% (614/1031)</td>
<td>57.69% (679/1177)</td>
</tr>
<tr>
<td>zlib/inftrees.c</td>
<td>100.00% (1/1)</td>
<td>99.25% (264/266)</td>
<td>98.17% (107/109)</td>
</tr>
<tr>
<td>zlib/uncompr.c</td>
<td>100.00% (2/2)</td>
<td>89.29% (50/56)</td>
<td>76.74% (33/43)</td>
</tr>
<tr>
<td>zlib/zutil.c</td>
<td>40.00% (2/5)</td>
<td>20.00% (9/45)</td>
<td>10.00% (3/30)</td>
</tr>
<tr>
<td>zlib_uncompress_fuzzer.cc</td>
<td>100.00% (1/1)</td>
<td>100.00% (8/8)</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>Totals</td>
<td>38.00% (19/50)</td>
<td>57.27% (1177/2055)</td>
<td>59.25% (986/1664)</td>
</tr>
</tbody>
</table>
Distribution Assumptions

- Make sure you use an appropriate statistical test, many results are not “normal”. A sign test may work best for comparing results of different tools.

Power

- Make sure you have enough tests to have valid statistical power
  - Say you want to say Tool A is 3% better than Tool B
    - What is the confidence interval? 3% +/- 1% with 95% confidence sounds good to me
    - Requires over 9,000 test cases for this confidence interval out of a population of 1 Million

Randomness

- The above power results assume your samples are randomly selected across the whole population. Not often the case with software.
CONCLUSION

• Look at the data
• Use large random data sets
• Use appropriate statistics (such as paired statistical test, eg. sign-test)
• Consult statistician
• If data will have small percentage of true conditions versus false conditions, ignore the true negatives
• Ask review committees to hold authors accountable on presenting results.
Thank you

Questions?

jimaf@uidaho.edu