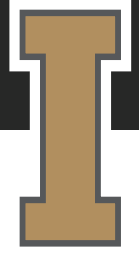
CYBSERSECURITY AT UNIVERSITY OF IDAHO

Jim Alves-Foss

University of Idaho



University of Idaho
College of Engineering

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

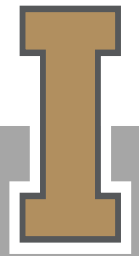
- Binary analysis, design and analysis of secure systems, industrial controls system security (specifically with respect to power grid), forensics, side-channel analysis, secure software engineering

Staff

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

History

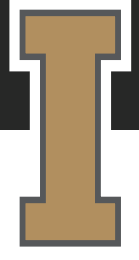
- Offered first cybersecurity course in 1992.
- NSA CAE since 1999. NSF Cybercorps SFS school since 2001.
- 6 CS cybersecurity faculty, 3 ECE cyber security faculty, plus others with interest.



PERCENTAGES, PROBABILITIES AND PROFESSIONS OF PERFORMANCE

Jim Alves-Foss

University of Idaho



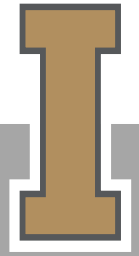
University of Idaho
College of Engineering

HOW WE REPORT EXPERIMENTAL CS/CYBER RESULTS

Several papers in computer science/cybersecurity compare the results of some tool or algorithm to:

- Other tools/algorithms
- Across different datasets

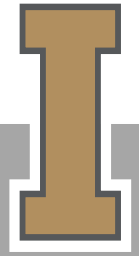
The papers are written to show successful results, but often with limited information and poor statistics.



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

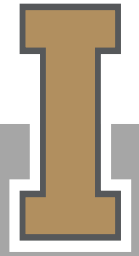
Reverse Engineering and Decompilation

Automated Analysis

- Unit level testing of binaries

Binary Patching

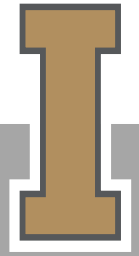
- Correcting security flaws (Goal of DARPA CGC and AMP programs)
- Inserting security protections
  - control flow integrity, stack guards



PRIOR ACADEMIC ATTEMPTS

Academic

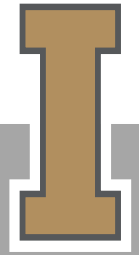
- Bao et al developed a machine-learning based approach
  - tested on Unix utilities for gcc and icc
  - appears that results incorporated into BAP
- Shin et al developed a neural network approach
  - tested on Unix utilities for gcc and icc
  - did not follow through on this
- Andriesse et al developed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



PRIOR COMMERICIAL ATTEMPTS

Commercial

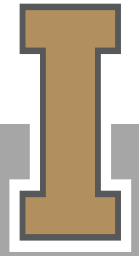
- IDA Pro automated process (tested with free version)
- NSA's Ghidra



HOW GOOD ARE PRIOR SOLUTIONS?

Following model of prior work used 3 datasets

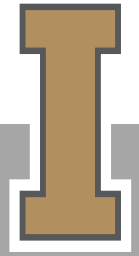
- Unix utilities: binutils, coreutils, findutils (129 unique programs)
- SPEC CPU 2017 (INTSPEED, INTRATE, FPSPEED, FPRATE)
  - Includes some Fortran code
  - Includes C/C++ with some exception handling
  - 28 unique 32-bit programs and 32 unique 64-bit programs
  - some require –fpic (position independent 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?

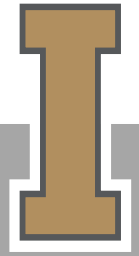


HOW DO WE MEASURE SUCCESS (2)?

Prior work uses the following:

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

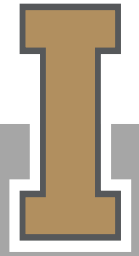
Your numbers are bigger/better than their numbers.

Win !!

Are you always better?

- If not, where and why?

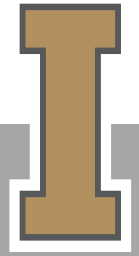
Are their a few instances where you are a lot better and these skew the means?



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?



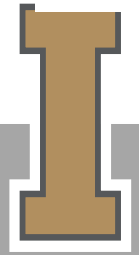
EXAMPLES PERCENTAGES…

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

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

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

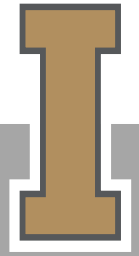
	gcc x86	gcc x64	clang x86	clang x64	VS x86	VS x64
IDA Pro 6.7	0.98/0.78	0.97/0.74	0.98/0.78	0.98/0.77	0.84/0.93	1.00/0.94
BAP/ByteWeight 0.9.9	0.68/0.83	0.70/0.66	0.52/0.71	0.73/0.49	0.63/0.74	0.69/0.56
Dyninst 9.1.0	0.93/0.91	0.96/0.74	0.98/0.95	0.88/0.72	–	–
Nucleus	0.98/0.96	0.98/0.96	0.96/0.97	0.96/0.95	0.86/0.96	0.95/0.94
△Nucleus	+0.00/+0.05	+0.01/+0.22	−0.02/ +0.02	−0.02/+0.18	+0.02/+0.03	−0.05/+0.00



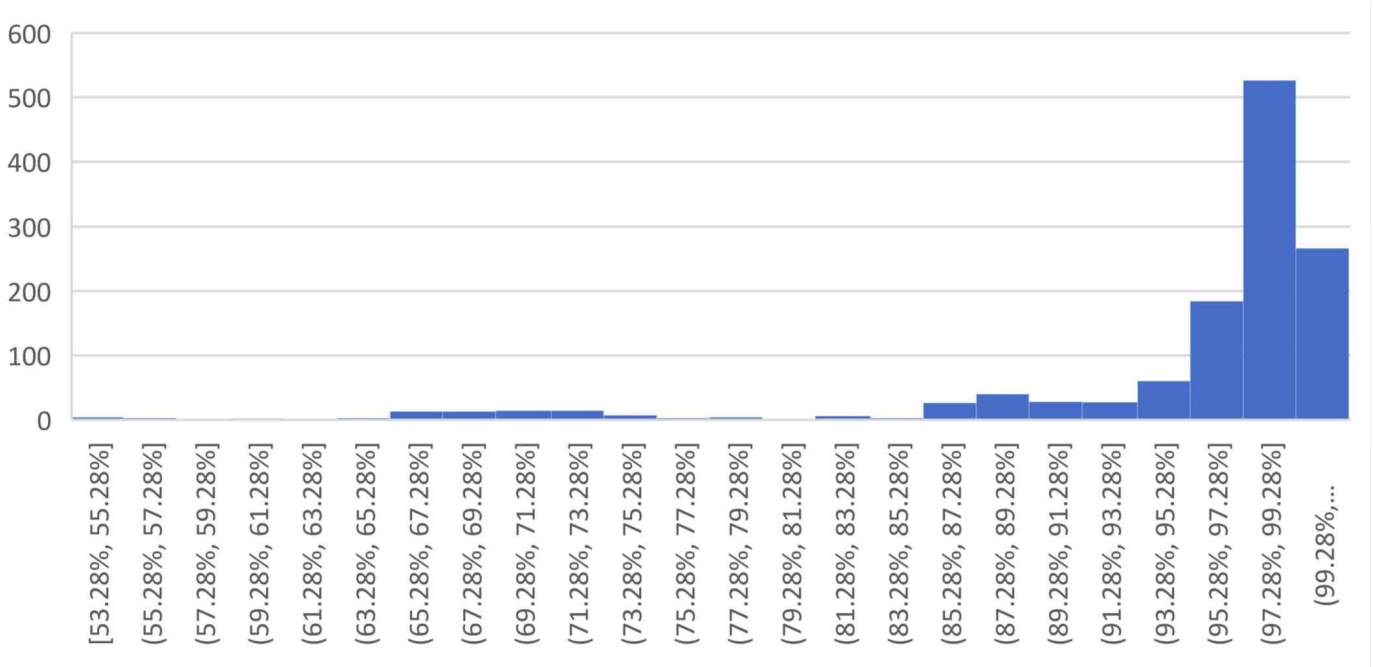
WHAT IS THE NATURE OF YOUR RESULTS

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

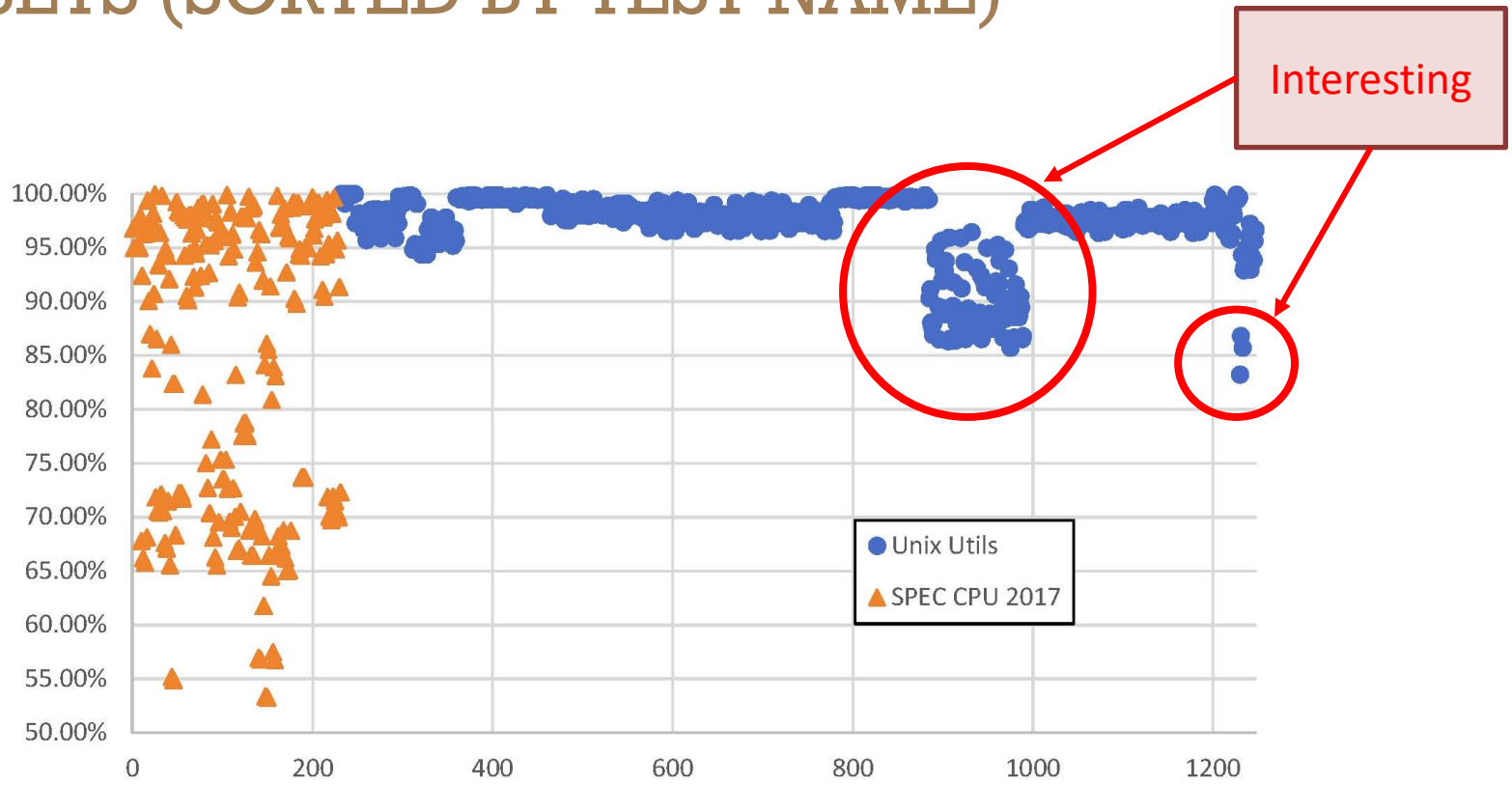
Visualize the Data



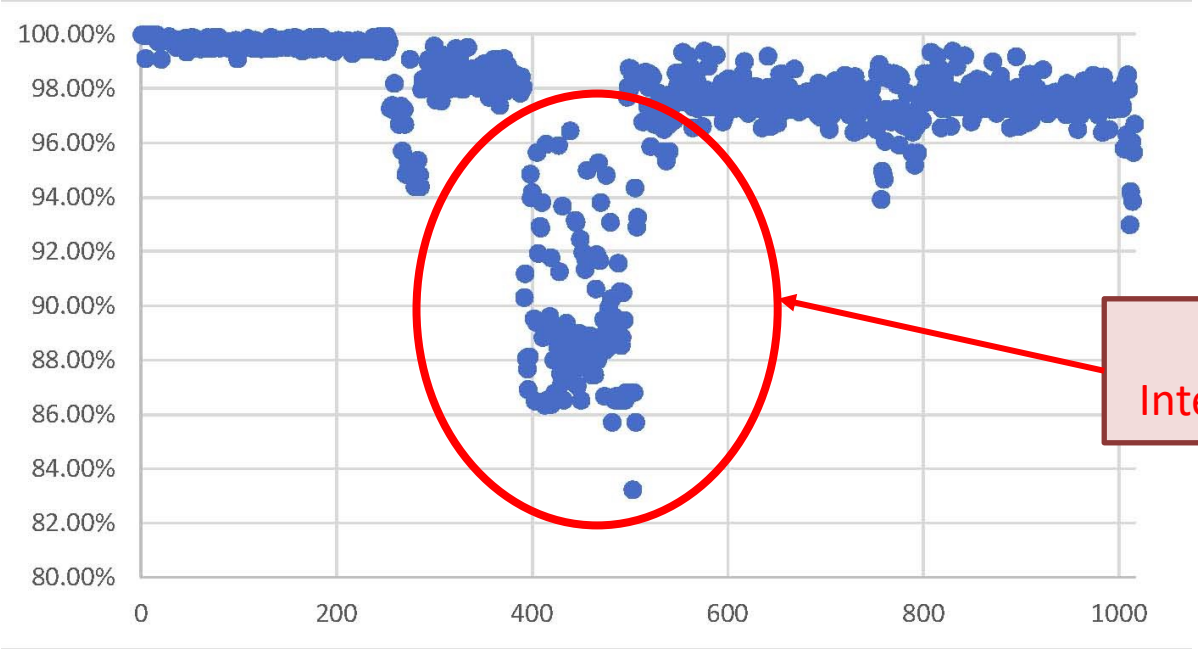
HISTOGRAM OF F1 VALUES FOR COMBINED DATASETS



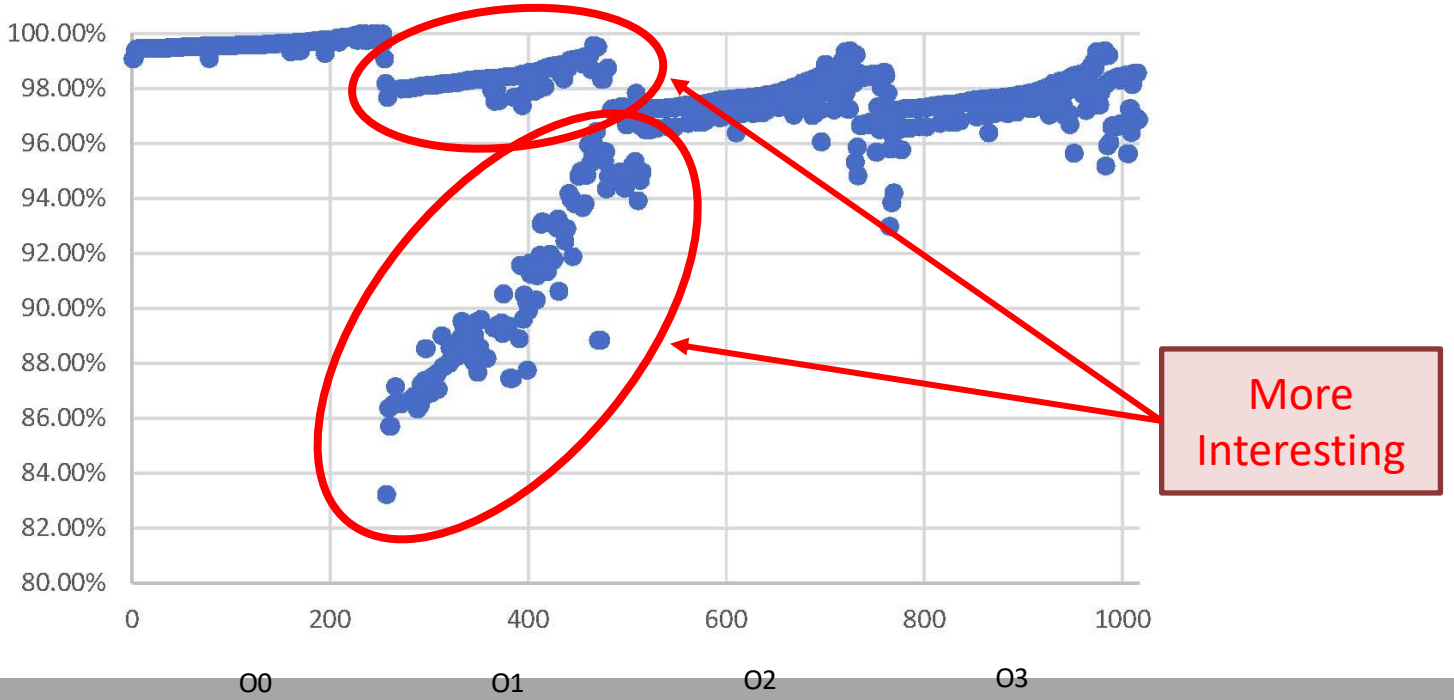
SCATTERPLOT OF F1 VALUES FOR COMBINED DATASETS (SORTED BY TEST NAME)



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



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



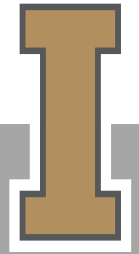
Now, lets look at the following visualizations when comparing tools for function boundary (specifically function start) identification. *(Recall the tables)*

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

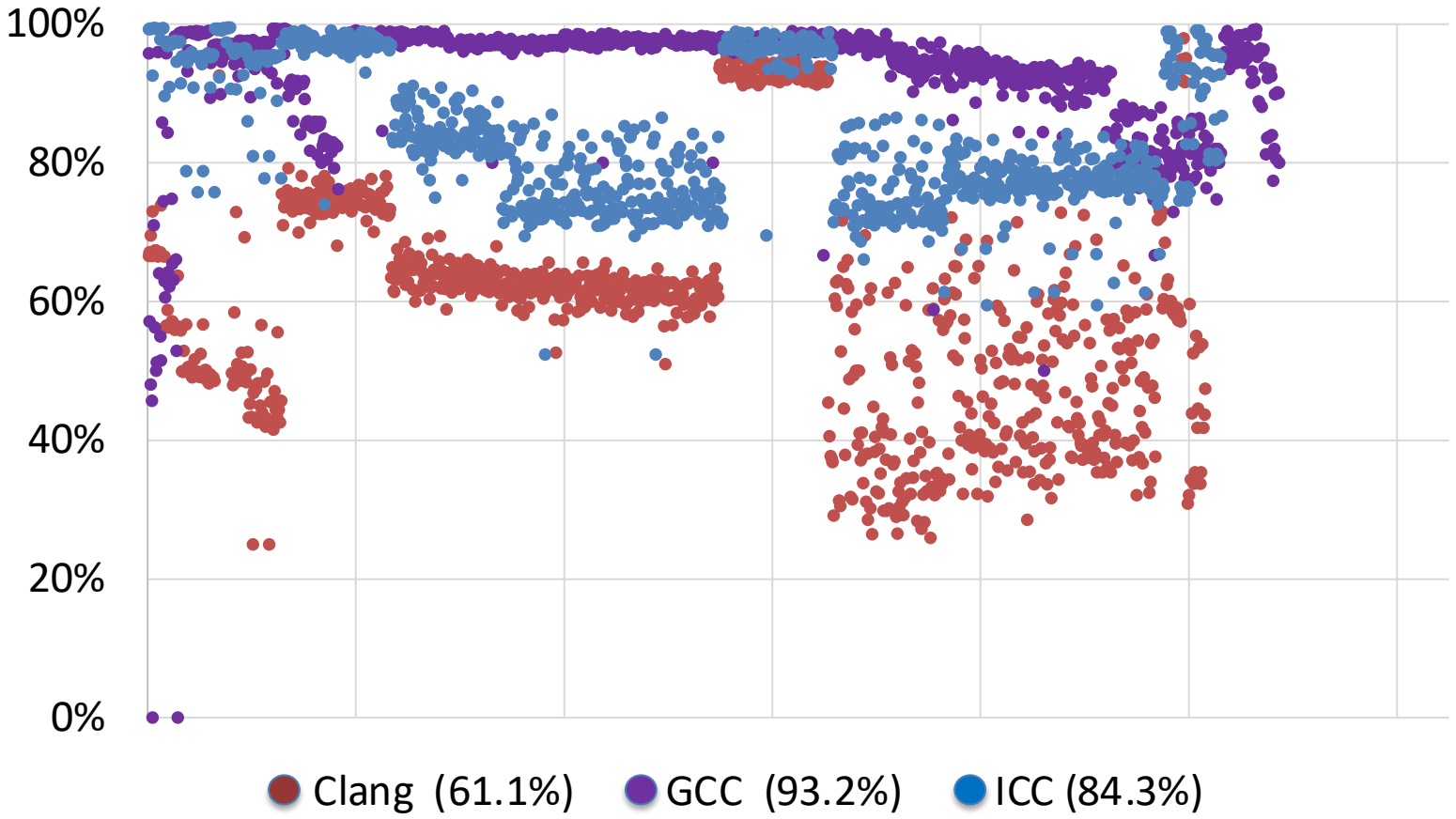
	GCC			ICC		
	Precision	Recall	Time(sec)	Precision	Recall	Time(sec)
Rosenblum et al. [7]	0.4909	0.4312	1172.41	0.6080	0.6749	2178.14
BYTEWEIGHT (3)	0.9103	0.8711	1417.51	0.8948	0.8592	1905.34
BYTEWEIGHT (no-norm)	0.9877	0.9302	19994.18	0.9727	0.9132	20894.45
BYTEWEIGHT	0.9726	0.9599	1468.75	0.9725	0.9800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

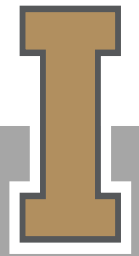
Table 2: Precision Recall of function start identification (reproduction of Table1(a) from Andriesse et al. [1])

	gcc x86	gcc x64	clang x86	clang x64	VS x86	VS x64
IDA Pro 6.7	0.98/0.78	0.97/0.74	0.98/0.78	0.98/0.77	0.84/0.93	1.00/0.94
BAP/ByteWeight 0.9.9	0.68/0.83	0.70/0.66	0.52/0.71	0.73/0.49	0.63/0.74	0.69/0.56
Dyninst 9.1.0	0.93/0.91	0.96/0.74	0.98/0.95	0.88/0.72	–	–
Nucleus	0.98/0.96	0.98/0.96	0.96/0.97	0.96/0.95	0.86/0.96	0.95/0.94
△Nucleus	+0.00/+0.05	+0.01/+0.22	−0.02/ +0.02	−0.02/+0.18	+0.02/+0.03	−0.05/+0.00

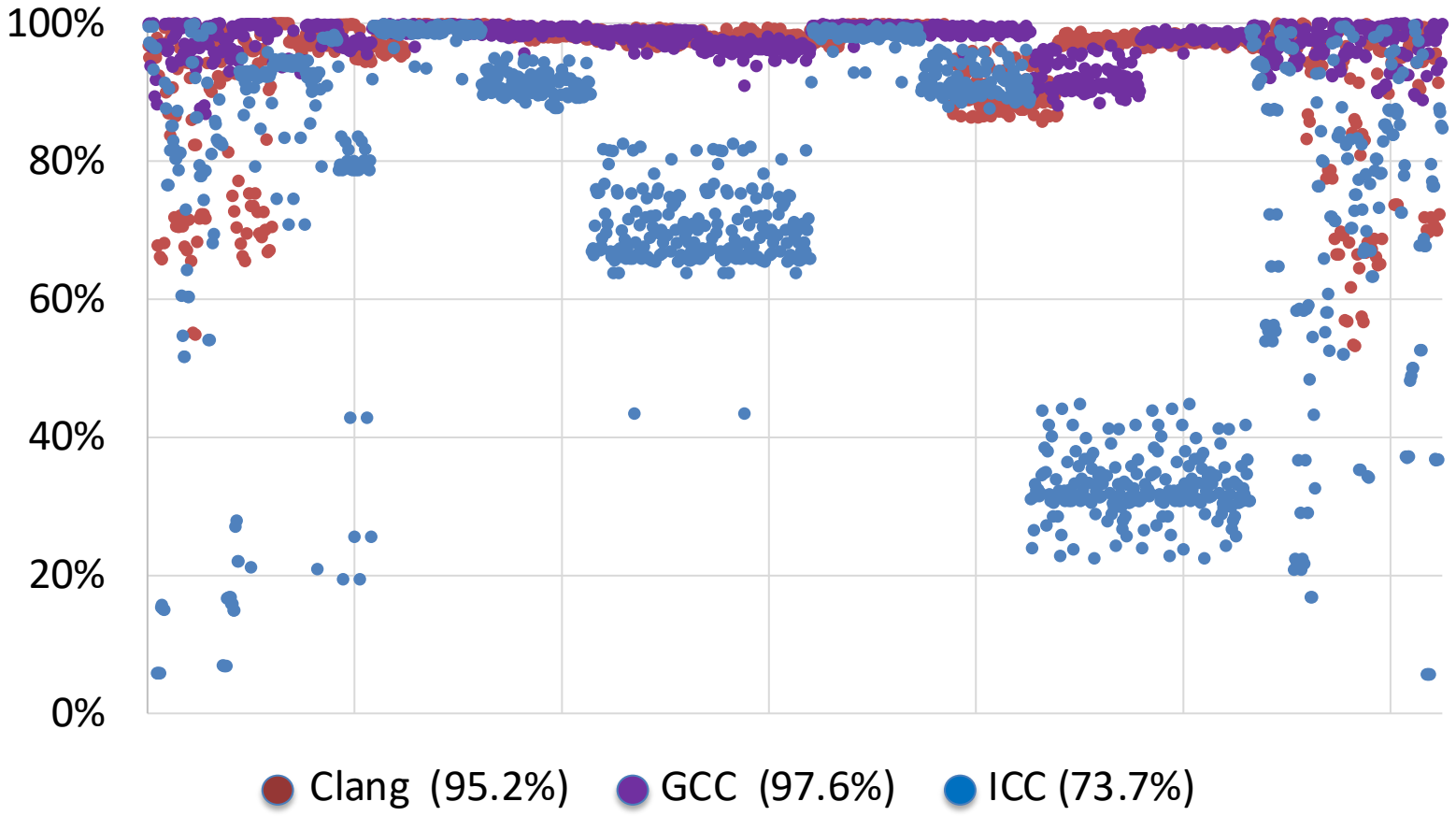


BAP - Function Starts (F1 values)

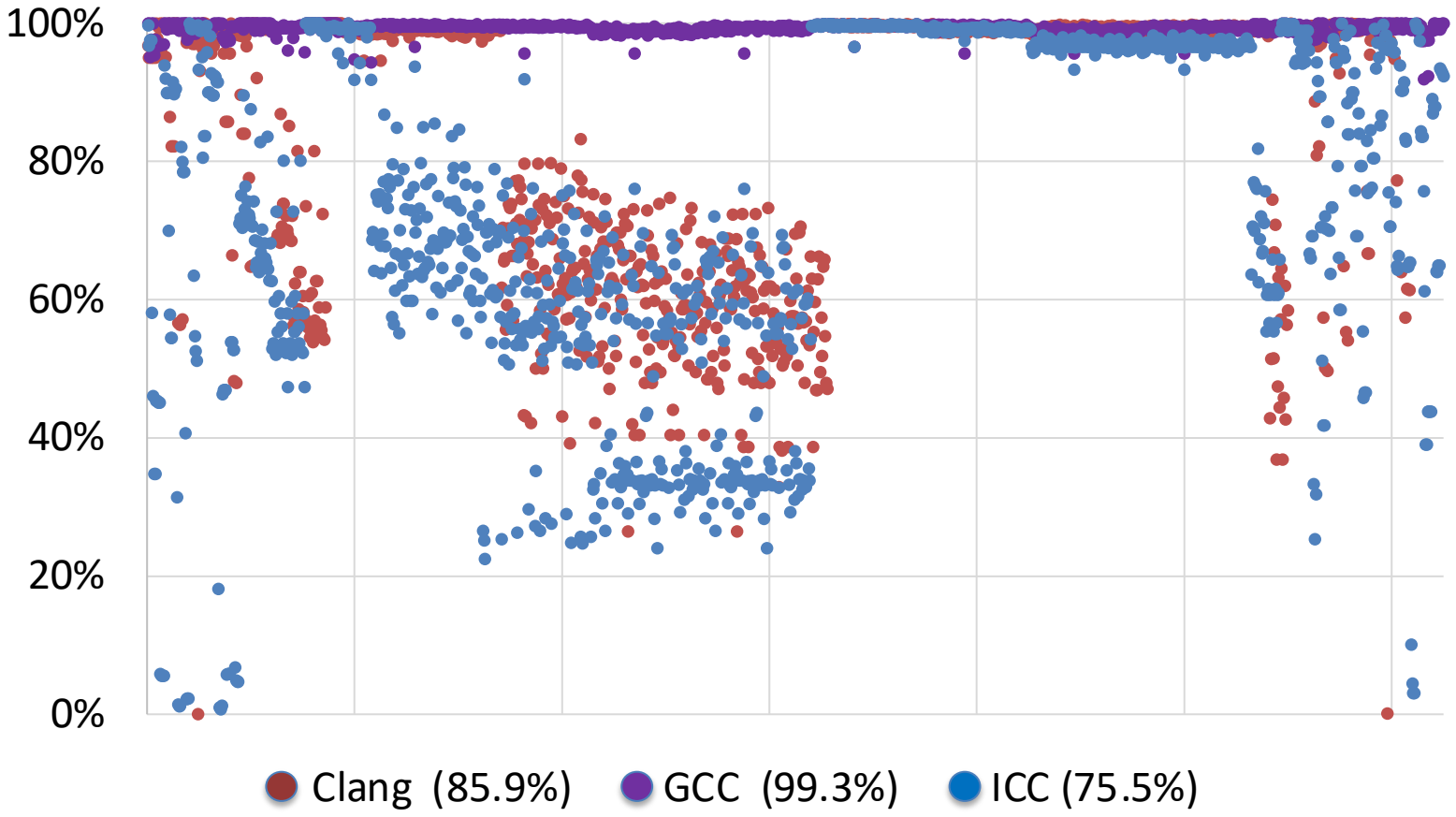


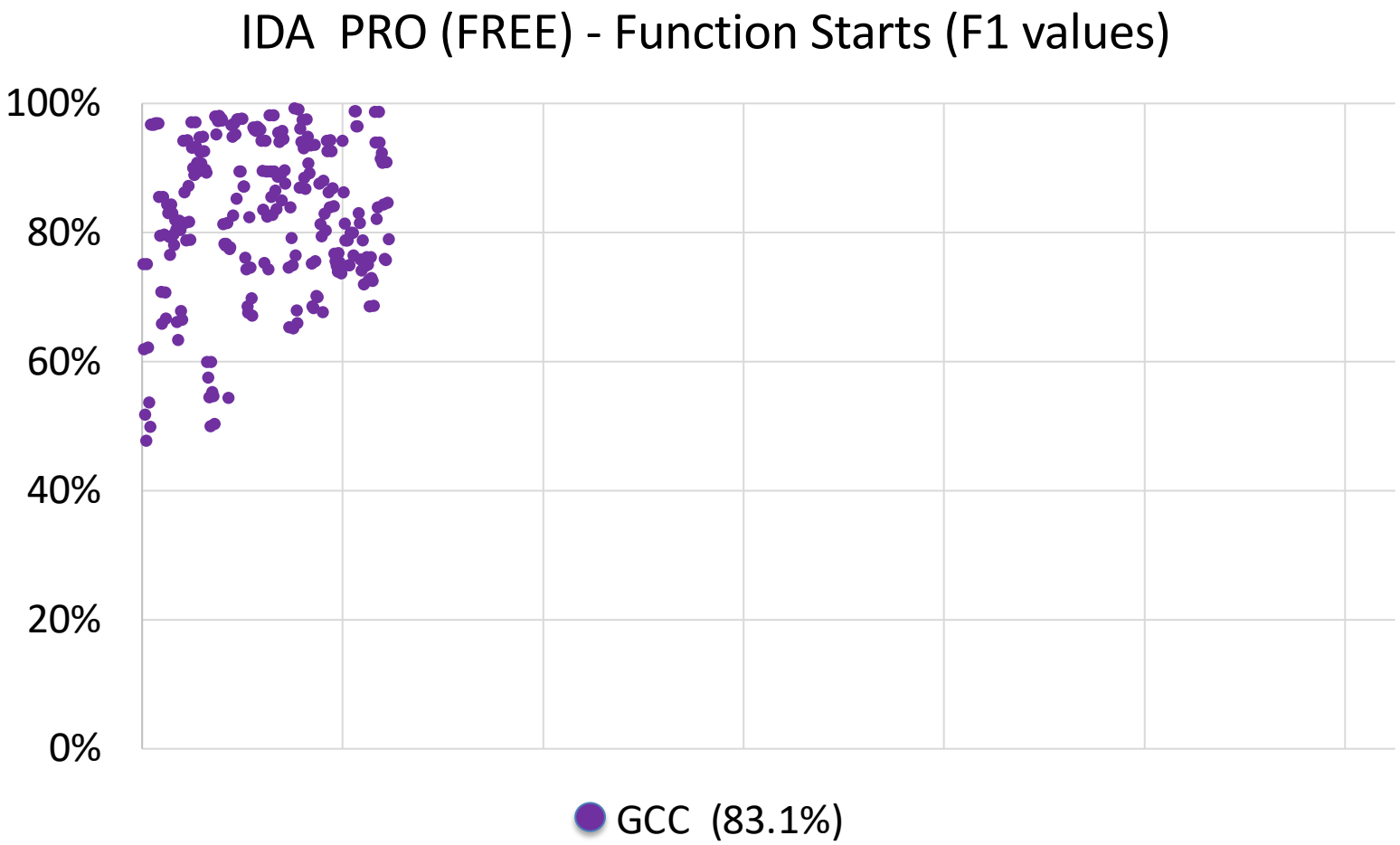


Nucleus - Function Starts (F1 values)

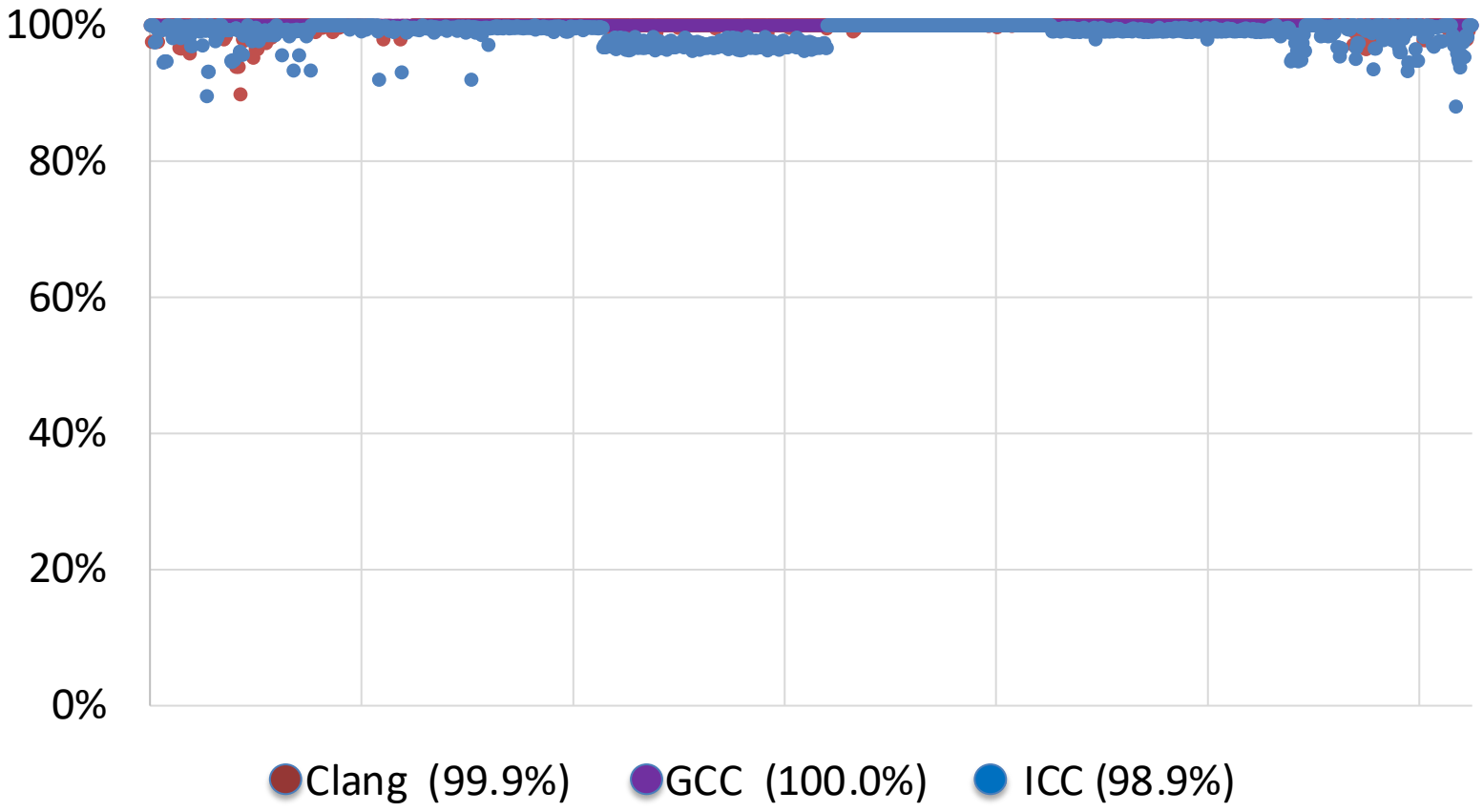


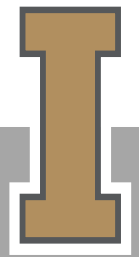
Ghidra - Function Starts (F1 values)



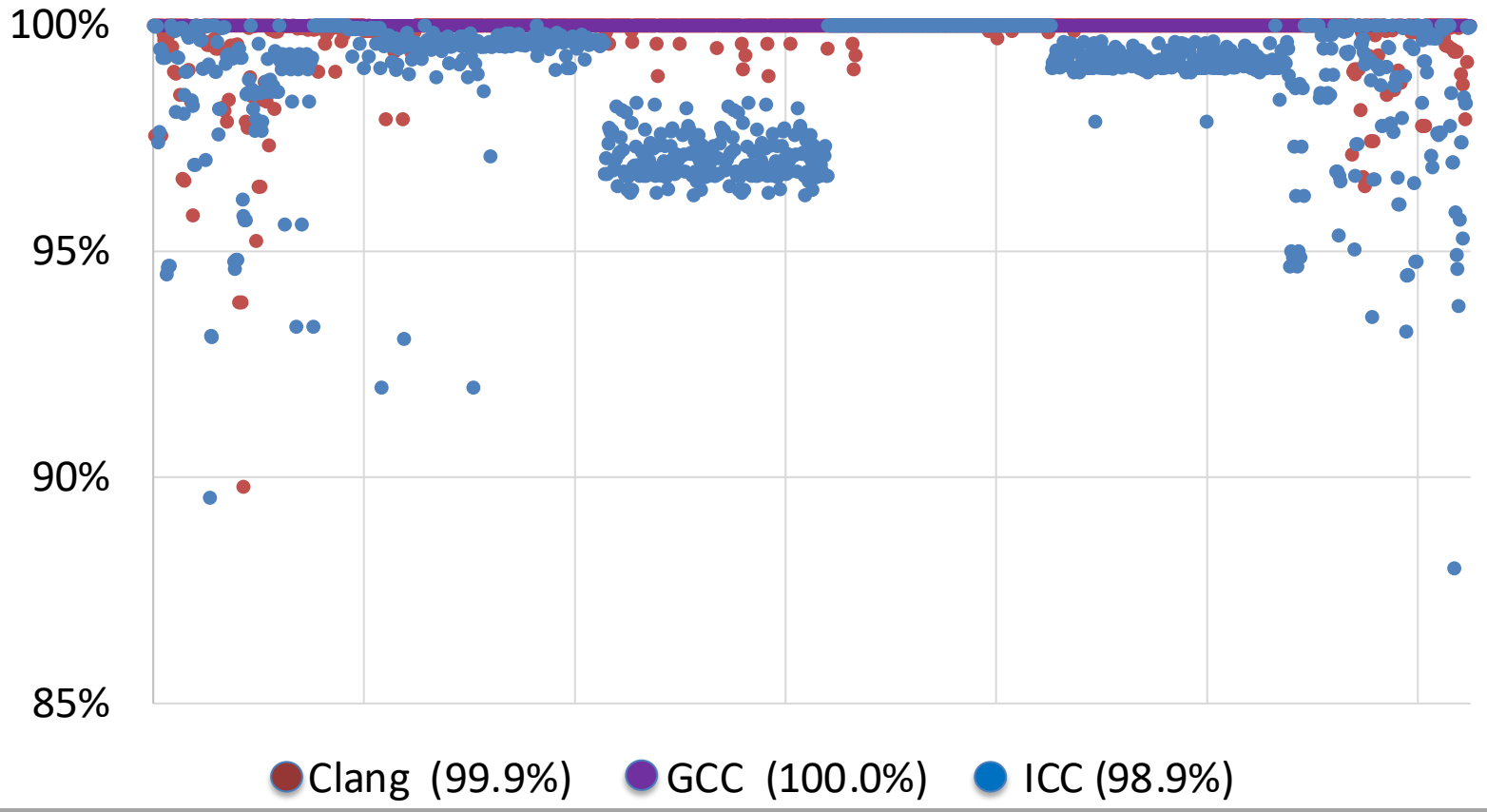

IDA PRO (FREE) - Function Starts (F1 values)
100%
80%
60%
40%
20%
0%
GCC (83.1%)

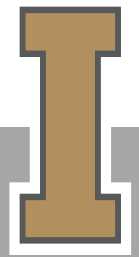
JIMA - Function Starts (F1 values) – as of Oct. 2019





JIMA - Function Starts (F1 values) – ZOOMED IN

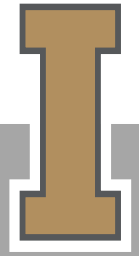




HOW ACCURATE IS ACCURACY?

	Condition Positive	Condition Negative
Predicted Positive	True Positive	False Positive (Type 1 error)
Predicted Negative	False Negative (Type 2 error)	True Negative

Confusion Matrix



SOME DEFINITION FROM CONFUSION MATRIX

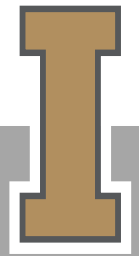
$$Prevalence = \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total Population}}$$

$$Accuracy = \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total Population}}$$

$$Precision = \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted positive}}$$

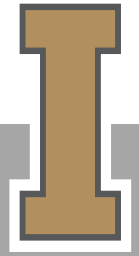
$$Recall = \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$



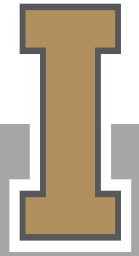
CONSIDER THE FOLLOWING **GROUND TRUTH** (WHERE THERE ARE RARE INSTANCES OF CONDITION WE ARE LOOKING FOR)

	Condition Positive	Condition Negative
Predicted Positive	10	
Predicted Negative		990



99% ACCURATE

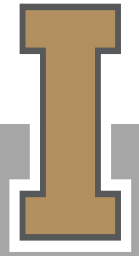
	Condition Positive	Condition Negative
Predicted Positive		
Predicted Negative	10	990



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

PRECISION, RECALL AND F1 VALUE

	Condition Positive	Condition Negative
Predicted Positive		
Predicted Negative	10	990



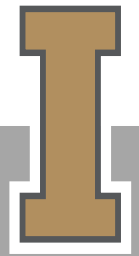
Precision is 0%, Recall is 0%, F1 is 0%

PRECISION, RECALL AND F1 VALUE (NEW EXAMPLE)

	Condition Positive	Condition Negative
Predicted Positive	5	5
Predicted Negative	5	985

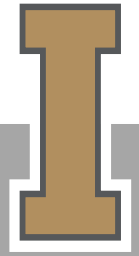
These areas are relevant

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



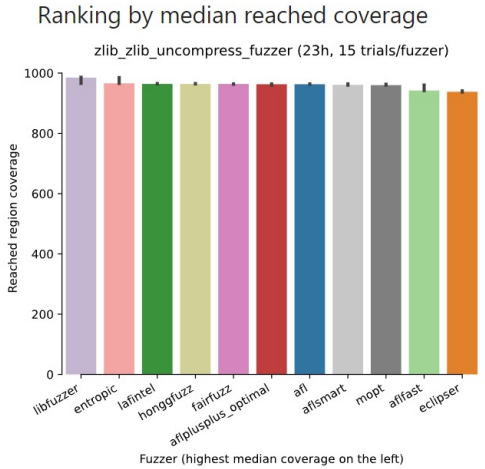
KNOW WHAT IS BEING MEASURED

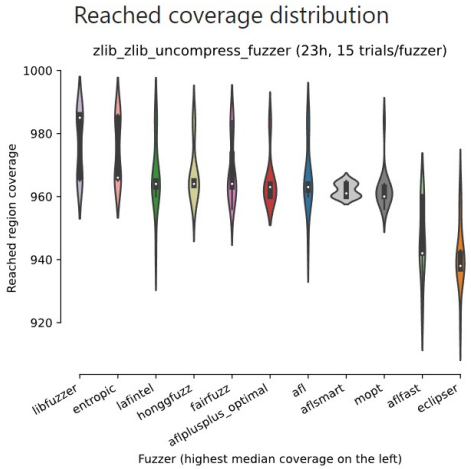
- Understand what you are measuring
- Understand the statistics
- Present it reliably

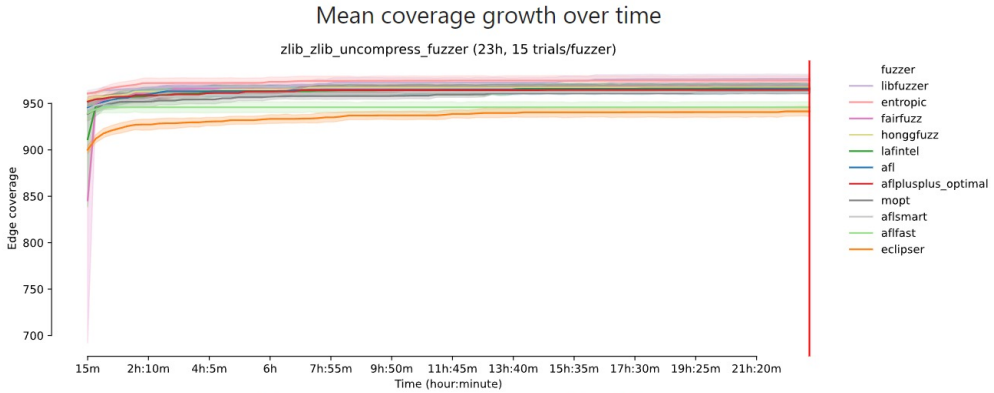


KNOW WHAT IS BEING MEASURED (2)

zlib_zlib_uncompress_fuzzer summary







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

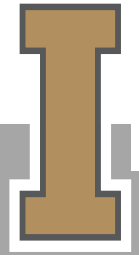
KNOW WHAT IS BEING MEASURED (3)

Coverage Report

Created: 2020-09-07 05:48

Click here for information about interpreting this report.

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



STATISTICAL RECOMMENDATIONS

Distribution Assumptions

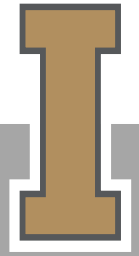
- Make sure you use an appropriate statistical test, many results are not “normal”. A sign test may work best for comparing results of different tools.

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

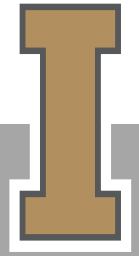
Randomness

- 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