Online Appendix for "FCEVAL: An Effective and Quantitative Platform for Evaluating Fuzzer Combinations Fairly and Easily"

Xiaoyun Zhao[a,1], Chao Yang[a,*], Zhizhuang Jia[a], Yue Wang[a] and Jianfeng Ma[a]

[a] *School of Cyber Engineering, Xidian University, No.266, Xifeng Road, Changan District, Xi'an, 710119, China*

ABSTRACT

Diverse base fuzzers collaborate as a fuzzer combination. Fuzzer combinations have been proven to perform more robustly and efficiently when fuzzing complex and diverse real-world programs. The efficiency of finding bugs with limited computational resources would benefit from the fuzzer combinations chosen by extensively quantitative performance evaluation. However, evaluating fuzzer combinations remains challenging due to the low efficiency of base fuzzers working collaboratively, and the absence of unified benchmarks, comprehensive metrics, and unified analysis methods for coverage and bugs. This prevents us from choosing efficient fuzzer combinations and thus hampers vulnerability mining in real-world targets. In this paper, we design and implement FCEVAL, an open-source platform for evaluating fuzzer combination evaluation. In detail, we propose a new sharing policies for test cases for effectiveness; select a unified set of diverse benchmarks and metrics while adopting unified methods of bug analysis and real-time independent analysis of branch coverage for fairness and quantification; and provide tools and guidelines covering the whole evaluation process for usability. Leveraging FCEVAL, we evaluate fuzzer combinations for more than 50,000 CPU hours, obtain a lot of precious test data, and come up with some important insights into evaluating fuzzer combinations. We do our best to promote the research and practice of collaborative fuzzing test with fuzzer combinations and give step-by-step help for all participants in fuzzing tests.

1. Introduction

This document provides additional material for the paper "FCEVAL: An Effective and Quantitative Platform for Evaluating Fuzzer Combinations Fairly and Easily".

2. Evaluation for FCC and FCD

We offer detailed evaluation results here for extensive observations.All experiment settings are identical to the evaluation for FCA and FCB except for benchmarks. We choose 6 programs from Binutils for assessment.

2.1. Answer to RQ1: Evaluating different test case sharing policies

2.1.1. Branch coverage

2.1.2. Bug discovery

2.2. Answer RQ2: Comprehensive quantitative assessment of different fuzzer combinations

2.2.1. Coverage quantity

We obtain details about the branch coverage as shown in Table 6.

2.2.2. Coverage speed

2.2.3. Bug quantity

2.2.4. Bug speed

2.3. Answering RQ3: Analyzing the correlation between coverage and bug

References

*Corresponding author

chaoyang@xidian.edu.cn (C. Yang)

ORCID(s):

Table 1
Total unique branches covered by FCA with the two sharing policies of test cases.

	cflow	jq	libpng	libxml2	lua	mp42aac	nm	objdump	openssl	readelf
$ENFUZZ^{1}$	686	994	1590	7621	2349	1262	7660	4360	1799	8237
$FCEVAL^{2}$	672	999	1592	7619	2325	1456	7455	4519	1812	8202
p-value	0.64	0.41	0.15	0.18	0.41	0.07	0.25	0.67	0.43	0.92
$\hat{A}12^{3}$	0.46	0.58	0.37	0.37	0.42	0.67	0.39	0.54	0.57	0.49

$ENFUZZ^{1}$ denotes the total number of unique branches covered with POLICY_ENFUZZ.
$FCEVAL^{2}$ denotes the total number of unique branches covered with POLICY_FCEVAL.
$\hat{A}12^{3}$:with POLICY_ENFUZZ as the baseline.

Table 2
Total unique bugs found by FCA with the two sharing policies of test cases.

	cflow	jq	libpng	libxml2	lua	mp42aac	nm	objdump	openssl	readelf
$ENFUZZ^{1}$	**16**	3	3	1	2	11	**688**	**14**	0	**8**
$FCEVAL^{2}$	15	3	3	**6**	**6**	11	674	11	0	3
p-value	0.26	1	0.76	0.42	0.67	0.13	0.21	0.21	1	0.89
$\hat{A}12^{3}$	0.40	**0.50**	**0.53**	**0.54**	**0.53**	**0.64**	0.38	**0.62**	**0.50**	0.49

$ENFUZZ^{1}$ denotes the total number of unique bugs discovered by FCA with POLICY_ENFUZZ.
$FCEVAL^{2}$ denotes the total number of unique bugs found by FCA with POLICY_FCEVAL.
$\hat{A}12^{3}$:with POLICY_ENFUZZ as the baseline.

Table 3
The detail of global branch coverage of the two fuzzer combinations.

Benchmark	FC	Total	Avg	p-value	$\hat{A}12$	Smax	Time (Minute)	Complementarity
cflow	FCB	**699**	171	-	-	181	1024	**399.22**
cflow	FCA	696	**175**	**0.06**	**0.89**	181	**200**	392.75
jq	FCB	**988**	**240**	-	-	248	**641**	529.23
jq	FCA	973	236	**0.06**	0.35	248	1314	**530.04**
libpng	FCB	1585	381	-	-	398	359	848.44
libpng	FCA	**1590**	**390**	**<0.01**	**0.88**	398	**273**	**849.66**
libxml2	FCB	7673	**1820**	-	-	1836	**1097**	**4008.04**
libxml2	FCA	**7710**	1801	**<0.01**	0.30	1836	1425	3978.16
lua	FCB	**2356**	**611**	-	-	646	**1114**	**1383.63**
lua	FCA	2350	595	**0.06**	0.31	646	1403	1346.73
mp42aac	FCB	**1528**	**246**	-	-	314	1429	**630.19**
mp42aac	FCA	1391	218	**0.04**	0.22	314	**1259**	576.70
nm	FCB	7401	1687	-	-	1827	1408	3823.16
nm	FCA	**7542**	**1736**	0.29	**0.60**	1827	**672**	**3945.75**
objdump	FCB	3606	706	-	-	990	1381	1738.29
objdump	FCA	**4826**	**940**	**<0.01**	**0.82**	990	**1024**	**2317.62**
openssl	FCB	1792	277	-	-	284	**1**	**659.06**
openssl	FCA	**1810**	277	0.82	**0.57**	284	**0**	640.84
readelf	FCB	7834	2484	-	-	2872	**1036**	5465.14
readelf	FCA	**8122**	**2686**	**<0.01**	**0.79**	2872	1431	**5793.16**

FC: fuzzer combination. Total: total unique branches covered. Avg: average number of covered branches.
$\hat{A}12$: with FCB as the baseline. Smax: smaller one of fuzzer combinations' maximum branches.

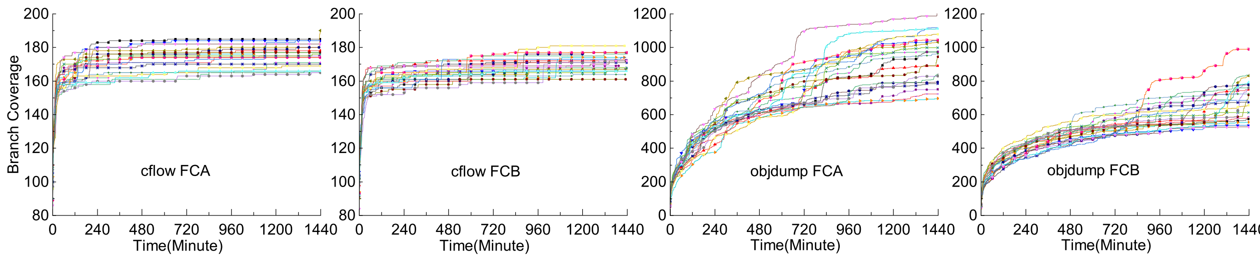

Figure 1: The branches covered by the two fuzzer combinations on cflow and objdump in 20 repeated times.

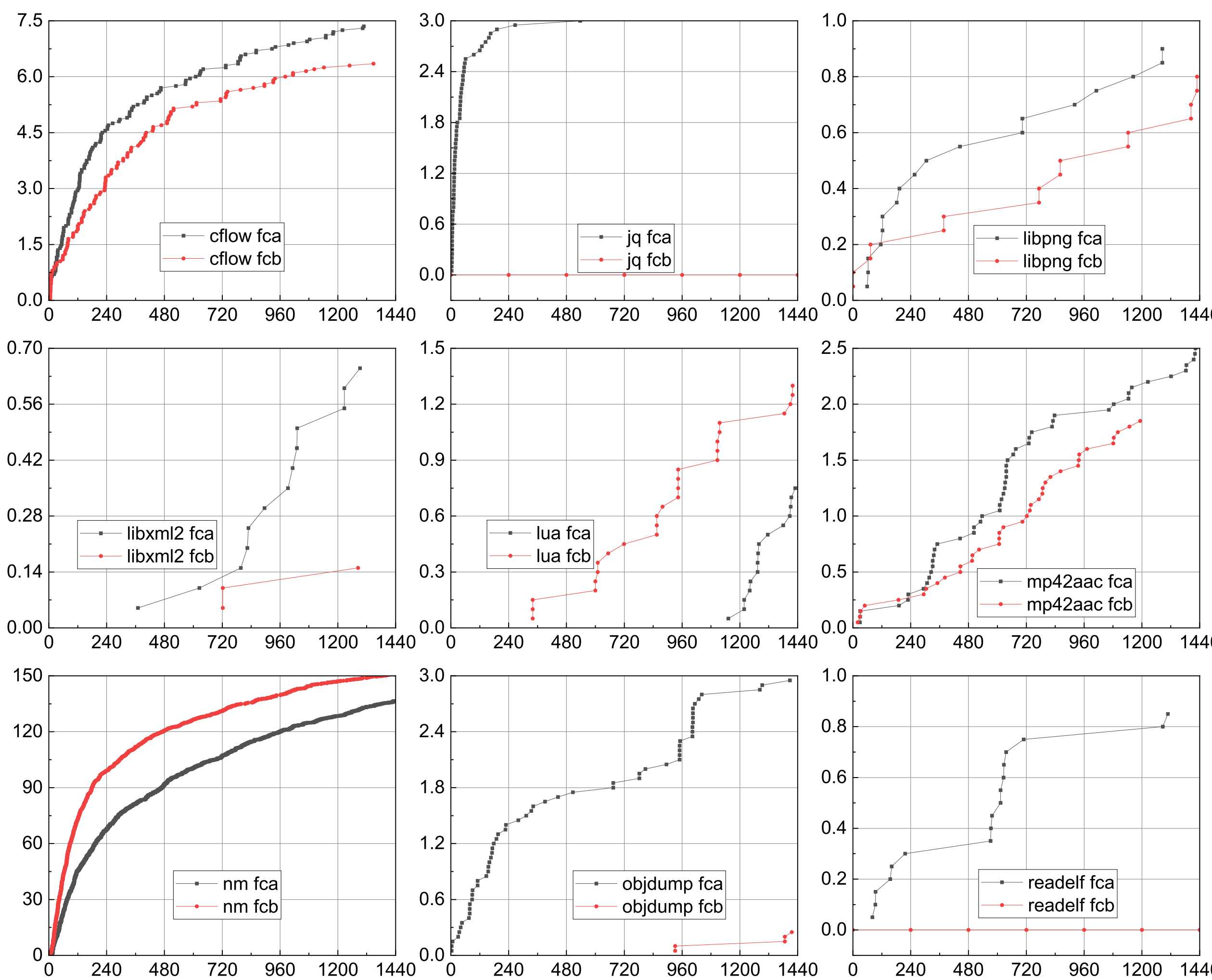

Figure 2: The average number of unique bugs found over time in 20 repetitions. eg. "cflow fca" shows the history of FCA on cflow.

Table 4
The detail of unique bugs found by the two fuzzer combinations in the 20 repeated experiments.

Benchmark	FC	Total	Avg	p-value	$\hat{A}12$	Faster
cflow	FCB	**18**	6.35	-	-	11
cflow	FCA	16	**7.40**	**0.02**	**0.71**	11
jq	FCB	0	0.00	-	-	0
jq	FCA	**3**	**3.00**	≤ 0.01	**1.00**	**3**
libpng	FCB	2	0.80	-	-	0
libpng	FCA	**3**	**0.90**	0.63	**0.54**	**3**
libxml2	FCB	3	0.15	-	-	3
libxml2	FCA	**7**	**0.65**	**0.05**	**0.65**	**7**
lua	FCB	**17**	**1.30**	-	-	**15**
lua	FCA	15	0.75	0.15	0.40	11
mp42aac	FCB	9	1.85	-	-	5
mp42aac	FCA	**13**	**2.90**	0.89	**0.51**	**8**
nm	FCB	**606**	**150.90**	-	-	**437**
nm	FCA	598	138.15	0.63	0.45	379
objdump	FCB	3	0.25	-	-	2
objdump	FCA	**20**	**3.25**	≤ 0.01	**0.96**	**20**
openssl	FCB	0	0.00	-	-	0
openssl	FCA	0	0.00	1.00	0.50	0
readelf	FCB	0	0.00	-	-	0
readelf	FCA	**8**	**1.00**	≤ 0.01	**0.73**	**8**

FC: fuzzer combination. Total: total number of unique bugs.
$\hat{A}12$: with FCB as the baseline.
Avg: average number of bugs found in repeated experiments.
Faster: number of bugs found faster by FCA or FCB.

Table 5
Spearman r_s for fuzzer combinations on different benchmarks.

FC	cflow	jq	libpng	libxml2	lua	mp42aac	nm	objdump	readelf
FCA	0.01	-	0.25	0.07	-0.19	0.81	0.37	0.49	0.08
FCB	-0.38	-	-0.12	0.34	0.31	0.46	0.35	0.22	-

FC: fuzzer combination.