




AIFORE: Smart Fuzzing Based on Automatic Input Format Reverse Engineering

Ji Shi, Zhun Wang, Zhiyao Feng, Yang Lan, Shisong Qin,
Wei You, Wei Zou, Mathias Payer, Chao Zhang

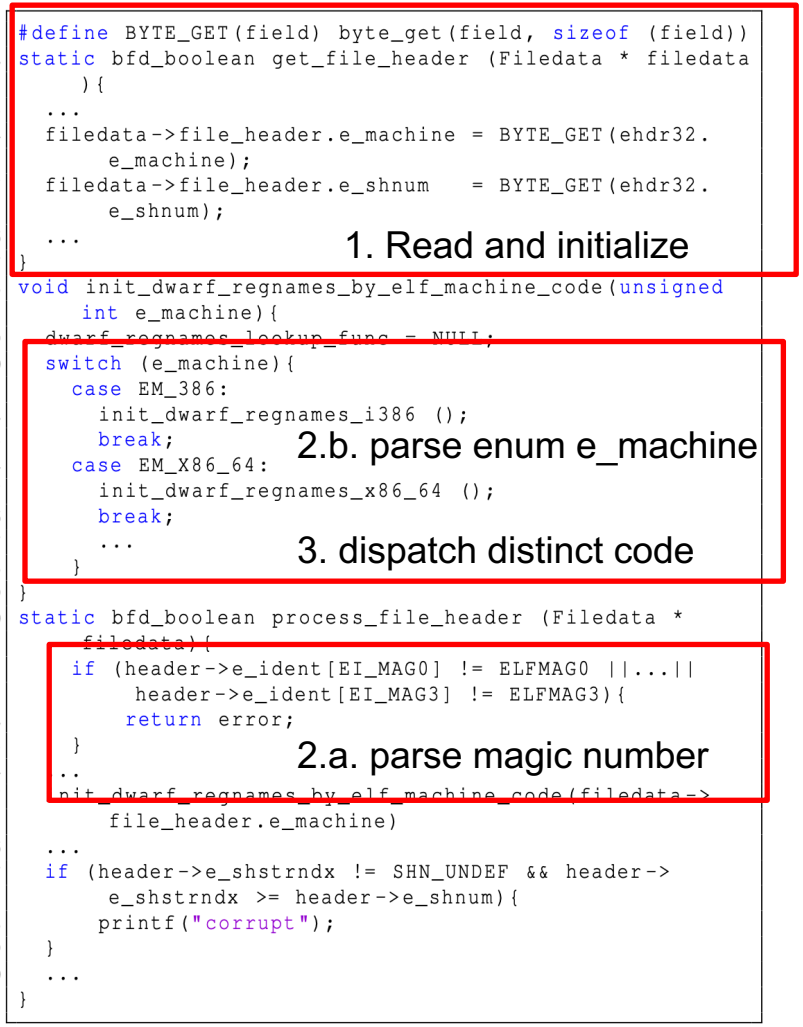
Presenter: Zhun
zhunwang.sigma@gmail.com

Recap: Fuzzing with Format

- **Fuzzing** produces a large number of inputs and feeds them to programs under test, detecting security violations.
- Require high quality (satisfy the format) of inputs, which enables efficient exploration of the program state space and increases the chance of triggering vulnerabilities.
- Knowledge of the **input format** is essential to generating high-quality inputs.

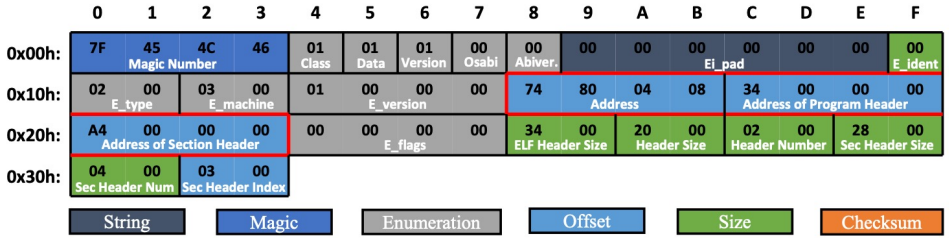
Motivational Example

- Observation 1: Bytes in an indivisible field are parsed together in one BB
- Observation 2: Programs processing fields of different types shows different patterns
- Observation 3: The structure of the input may differ, and the program will dispatch distinct code to parse the input

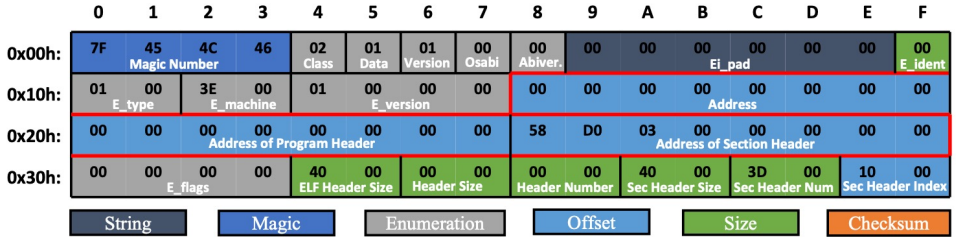


Motivation Example

- Observation 3: The structure of the input may differ, and the program will dispatch distinct code to parse the input



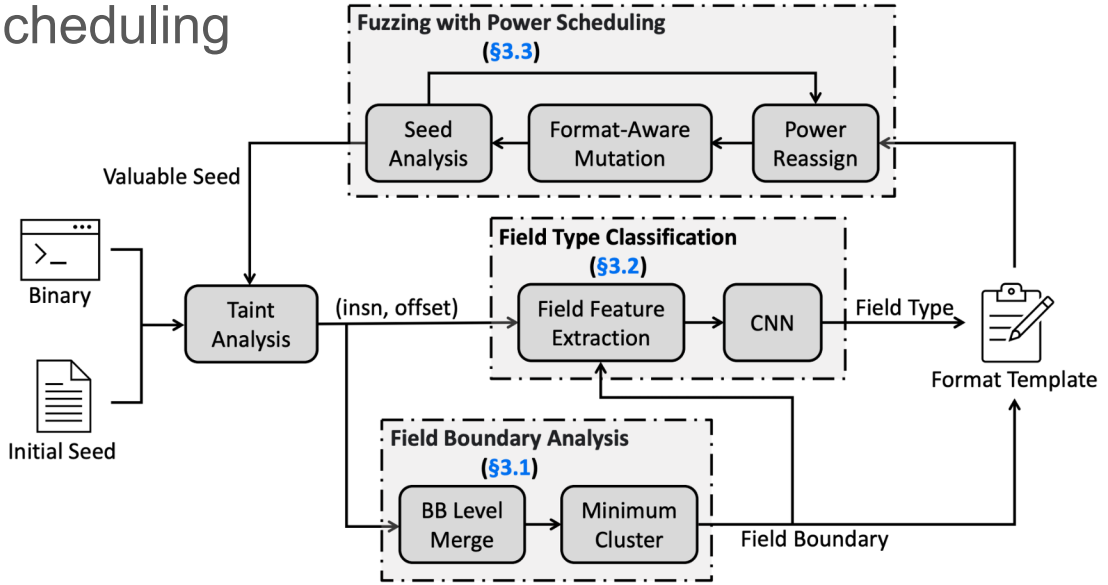
(a) ELF file header structure of 32bit.



(b) ELF file header structure of 64bit.

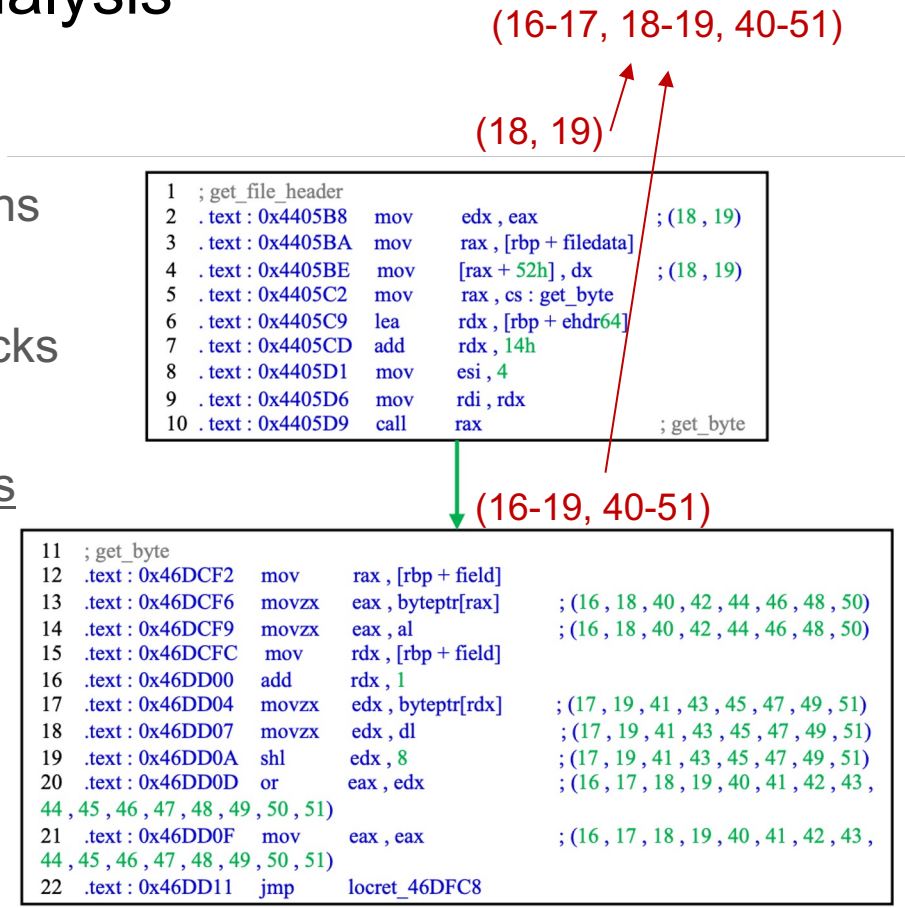
AIFORE: Overview

- Basic-block-based Format Extraction
  - (B) Field Boundary Analysis
  - (T) Field Type Classfification
- (P) Format-based Power Scheduling



AIFORE: B - Field Boundary Analysis

- Taint Analysis
  - Mapping input bytes and instructions
- Basic Block Level Merge
  - Mapping input bytes and basic blocks
- Minimum Cluster Method
  - Get minimum units as format Fields
  - Mapping fields and basic blocks



AIFORE: B - Field Boundary Analysis

- Taint Analysis
- Basic Block Level Merge
- Minimum Cluster Method

7F	45	4C	46	01	01	01	00	00	00	00	00	00	00	00	00
02	00	03	00	01	00	00	00	74	80	04	08	34	00	00	00
A4	00	00	00	00	00	00	00	34	00	20	00	02	00	28	00
04	00	03	00												

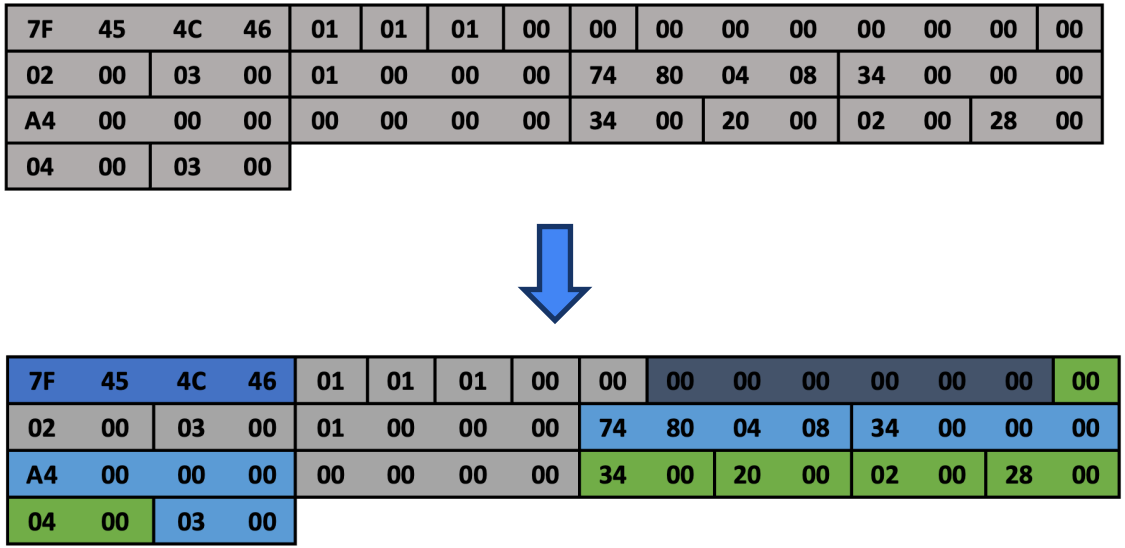
7F	45	4C	46	01	01	01	00	00	00	00	00	00	00	00	00
02	00	03	00	01	00	00	00	74	80	04	08	34	00	00	00
A4	00	00	00	00	00	00	00	34	00	20	00	02	00	28	00
04	00	03	00												

AIFORE: T - Field Type Classification

- -> Mapping fields and basic blocks
- Field Feature Extraction
  - Data vectorization with one-hot encoding
  - *IR features*: lifting instructions to IRs, then record related IRs
  - *Format string features*: ‘%x’ - integer, ‘%s’ - string
  - *Library call features*: record special function calls in BB, ‘memcpy’, ‘strcpy’, ‘malloc’, etc.
- CNN Classification
  - Semantic Types: Size, Enumeration, Magic Number, String, Checksum, Offset

AIFORE: T - Field Type Classification

- Field Feature Extraction
- CNN Classification



AIFORE: P - Format-based Power Scheduling

- Power Scheduling for Format Analysis
  - Which seed should we perform format extraction on?
- Power Scheduling for Mutation
  - How can we balance the fuzzing power for different format variants produced during mutation?

AIFORE: P - Format-based Power Scheduling

- Power Scheduling for Format Analysis
  - Which seed should we perform format extraction on?
  - On Valuable seeds: reach more new BBs and likely belong to an unseen format variant
- Power Scheduling for Mutation
  - How can we balance the fuzzing power for different format variants produced during mutation?
  - Utilize custom mutators for different types of fields
  - Re-assign power to those seeds bound with less mutated format variants and skip those bound with the fully mutated format variants

Implementation

- Taint Analysis Engine
  - VUzzer, libdft
- IR Lifting and Basic Block Extraction
  - Angr
- CNN Components
  - Keras
- Format Ground Truth Data
  - 010Editor templates
- Fuzzing
  - AFL

libdft

K Keras

Evaluation

- Format Extraction Performance
  - Accuracy and Time performance
  - Performance under different configs
    (compiler optimization level, amount of training data, seed size)
- Fuzzing Performance
  - Code coverage
  - Bug detection
  - Contribution of each module

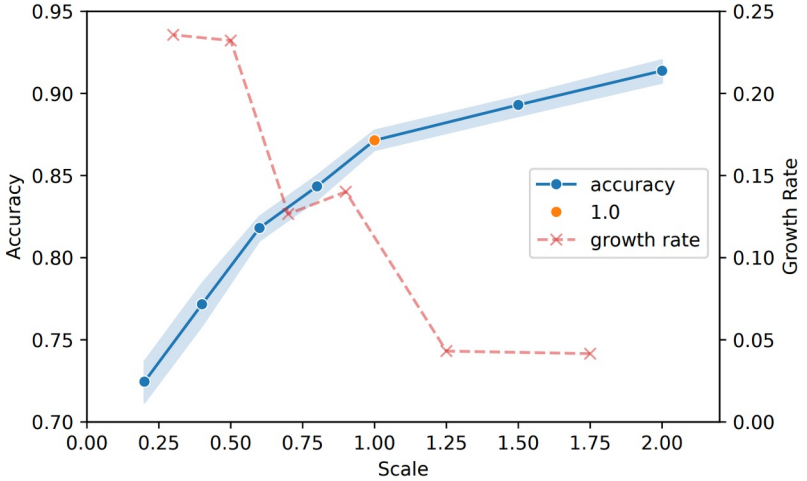
Evaluation: Field Boundary Accuracy - Opt. levels

- The accuracy for different optimization levels of target programs
- The accuracy is irrelevant to the compiler optimization level

Format	Program	Avg. Field Count	Avg. Size (bytes)	Optimation Levels				
				-O0	-O1	-O2	-O3	-Os
7Z	7za	9	425	55.56%	55.56%	55.56%	55.56%	55.56%
BMP	jasper	1,243	1,702	88.81%	89.38%	88.58%	89.14%	89.14%
ELF	readelf	77	324	96.10%	93.51%	93.51%	93.51%	93.51%
GIF	gifsicle	217	821	96.77%	96.77%	97.23%	97.23%	94.00%
JPG	exiv2	24	424	50.00%	50.00%	55.00%	53.58%	50.00%
OTF	freetype_parser	258	226,772	75.44%	76.64%	61.06%	61.06%	58.58%
PCAP	tcpdump	22	114	90.90%	90.90%	90.90%	88.64%	86.36%
PNG	pngtest	42	239	63.12%	64.31%	63.12%	63.12%	63.12%
TIFF	tiffdump	73	22,700	82.19%	82.19%	76.71%	76.71%	82.19%
TTF	freetype_parser	110	7,876	76.36%	76.36%	76.36%	76.36%	76.36%
WAV	sfinfo	18	37,524	77.77%	72.22%	72.22%	72.22%	72.22%
XLS	xls2csv	288	19,968	41.67%	41.67%	36.81%	36.81%	36.81%
ZIP	7za	12	906	58.33%	58.33%	58.33%	58.33%	58.33%
Average		**184**	**24,600**	**73.31%**	**72.91%**	**71.18%**	**70.94%**	**70.48%**

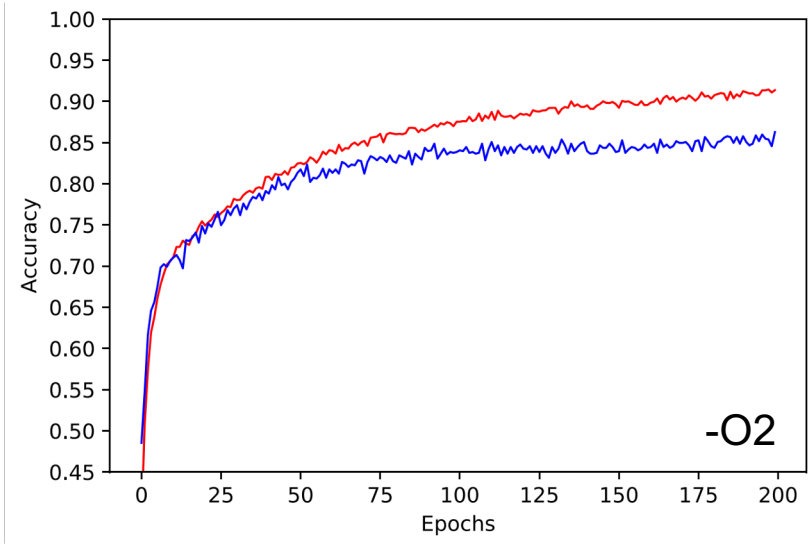
Evaluation: CNN Performance - data amount

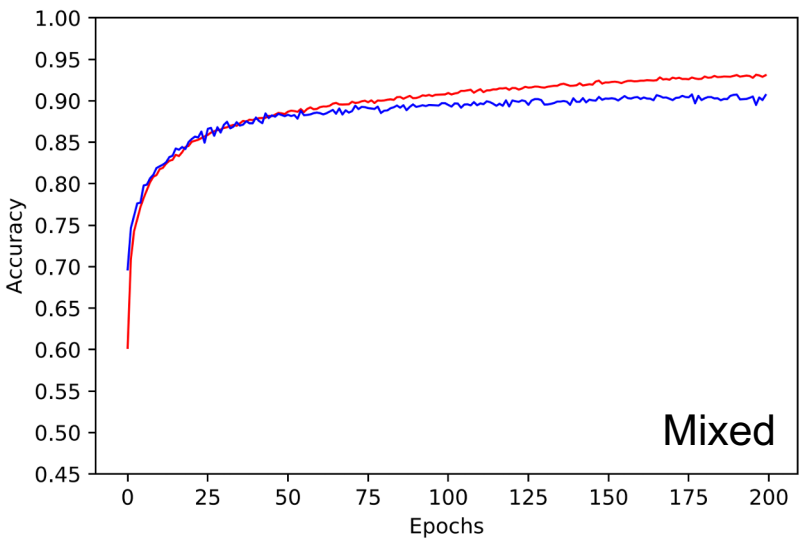
- **Data amount** requirement to train a sufficiently reliable CNN model
- Mean validation accuracy and growth rate among different scales of training set. Set 10k fields as unit 1.0
- A larger amount of training data results in higher model accuracy.
- However, the growth rate stagnates quickly when the scale is larger than 1.0.



Evaluation: CNN Performance - Opt. level

- CNN model performance on the training set and validation set
- Accuracy on training/validation set with -O2 and mixed optimization levels





Evaluation: Field Type Accuracy

- The accuracy for different optimization levels and unseen formats
- Upper half: unseen formats for trained programs
- Lower half: unseen formats for untrained programs

top-1 acc. / top-2 acc.

Program	Format	Samples	Optimization Levels					
			-O0	-O1	-O2	-O3	-Os	Mixed
exiv2	GIF	202	73.20%/80.00%	84.62%/100.00%	100.00%/100.00%	94.12%/100.00%	100.00%/100.00%	95.14%/100.00%
exiv2	JPG	109	77.78%/88.89%	90.00%/100.00%	66.67%/77.78%	80.00%/80.00%	72.73%/81.82%	76.66%/85.28%
freetype_parser	OTF	361	74.13%/86.42%	71.30%/85.19%	76.47%/87.50%	80.00%/90.00%	67.86%/89.29%	70.56%/92.23%
7za	ZIP	364	89.01%/96.98%	93.37%/97.79%	94.90%/99.06%	91.23%/98.25%	92.99%/97.19%	91.70%/98.18%
Average			**78.53%/88.07%**	**84.82%/95.75%**	**84.51%/91.09%**	**84.51%/92.06%**	**83.40%/92.07%**	**83.52%/93.92%**
elfutils-readelf	ELF	5385	62.66%/74.59%	73.93%/78.67%	61.11%/87.50%	54.99%/64.33%	71.69%/72.21%	67.33%/75.77%
jasper	BMP	1068	83.11%/97.30%	92.00%/99.43%	90.06%/99.42%	91.98%/100.00%	90.48%/98.10%	93.46%/100.00%
jhead	JPG	913	84.00%/88.00%	86.36%/86.36%	86.36%/90.91%	86.96%/86.96%	84.00%/92.00%	85.92%/86.92%
tcpdump	PCAP	1412	74.70%/91.42%	84.13%/85.72%	78.04%/91.77%	89.81%/91.40%	75.44%/81.43%	73.04%/85.83%
pngtest	PNG	1004	80.12%/87.77%	86.88%/91.10%	82.35%/85.88%	88.34%/89.11%	83.31%/88.96%	80.11%/82.31%
sfinfo	WAV	434	88.89%/100.00%	97.91%/100.00%	97.37%/100.00%	98.80%/98.80%	98.99%/100.00%	97.92%/99.74%
xls2csv	XLS	1339	71.00%/84.93%	75.36%/81.88%	68.00%/71.20%	71.94%/71.94%	65.56%/77.46%	66.95%/71.76%
Average			**77.78%/89.14%**	**85.22%/89.02%**	**80.47%/89.53%**	**83.26%/86.08%**	**81.35%/87.17%**	**80.68%/86.05%**

Evaluation: Field Type Accuracy

- The accuracy for different optimization levels and unseen formats
- AIFORE can predict the field type in unseen formats with high accuracy, while the Top-1 accuracy is over 80% and the Top-2 accuracy is over 85%
- the model performance is irrelevant to the programs' optimization levels

top-1 acc. / top-2 acc.

Program	Format	Samples	Optimization Levels					
			-O0	-O1	-O2	-O3	-Os	Mixed
exiv2	GIF	202	73.20%/80.00%	84.62%/100.00%	100.00%/100.00%	94.12%/100.00%	100.00%/100.00%	95.14%/100.00%
exiv2	JPG	109	77.78%/88.89%	90.00%/100.00%	66.67%/77.78%	80.00%/80.00%	72.73%/81.82%	76.66%/85.28%
freetype_parser	OTF	361	74.13%/86.42%	71.30%/85.19%	76.47%/87.50%	80.00%/90.00%	67.86%/89.29%	70.56%/92.23%
7za	ZIP	364	89.01%/96.98%	93.37%/97.79%	94.90%/99.06%	91.23%/98.25%	92.99%/97.19%	91.70%/98.18%
Average			**78.53%/88.07%**	**84.82%/95.75%**	**84.51%/91.09%**	**84.51%/92.06%**	**83.40%/92.07%**	**83.52%/93.92%**
elfutils-readelf	ELF	5385	62.66%/74.59%	73.93%/78.67%	61.11%/87.50%	54.99%/64.33%	71.69%/72.21%	67.33%/75.77%
jasper	BMP	1068	83.11%/97.30%	92.00%/99.43%	90.06%/99.42%	91.98%/100.00%	90.48%/98.10%	93.46%/100.00%
jhead	JPG	913	84.00%/88.00%	86.36%/86.36%	86.36%/90.91%	86.96%/86.96%	84.00%/92.00%	85.92%/86.92%
tcpdump	PCAP	1412	74.70%/91.42%	84.13%/85.72%	78.04%/91.77%	89.81%/91.40%	75.44%/81.43%	73.04%/85.83%
pngtest	PNG	1004	80.12%/87.77%	86.88%/91.10%	82.35%/85.88%	88.34%/89.11%	83.31%/88.96%	80.11%/82.31%
sfinfo	WAV	434	88.89%/100.00%	97.91%/100.00%	97.37%/100.00%	98.80%/98.80%	98.99%/100.00%	97.92%/99.74%
xls2csv	XLS	1339	71.00%/84.93%	75.36%/81.88%	68.00%/71.20%	71.94%/71.94%	65.56%/77.46%	66.95%/71.76%
Average			**77.78%/89.14%**	**85.22%/89.02%**	**80.47%/89.53%**	**83.26%/86.08%**	**81.35%/87.17%**	**80.68%/86.05%**

Evaluation: Format Extraction Accuracy - SOTA comparison

- Field boundary/type accuracy comparison between different input format reverse engineering solutions
- AIFORE achieves higher accuracy both in field boundary recognition and field type prediction

Format		Program	Large Seeds				
			Size(bytes)	AIFORE	ProFuzzer	AFL-Analyze	TIFF-fuzzer
Trained	ELF	readelf	808	96.43%/87.73%	37.40%66.25%	43.73%/40.00%	94.30%/(N/A)
	GIF	gifsicle	695	97.64%/87.31%	12.59%/52.94%	6.44%/11.76%	71.96%/(N/A)
	TIFF	tiffdump	448	82.23%/89.33%	27.13%/42.11%	13.19%/21.05%	81.30%/(N/A)
	TTF	freetype	542	67.43%/83.22%	30.28%/65.69%	9.93%/20.44%	5.56%/(N/A)
Untrained	PCAP	tcpdump	894	88.64%/82.34%	28.84%/71.14%	0.00%/39.04%	81.20%/(N/A)
	WAV	sfinfo	572	100.00%/95.55%	63.85%/66.67%		
	BMP	jasper	630	45.34%/83.54%	12.11%/91.18%		
	XLS	xls2csv	6656	81.25%/74.56%	(N/A)/(N/A)*		
Average			**1406**	**82.37%/85.45%**	**26.53%/57.00%**		

boundary acc. / type acc.

Small Seeds				
Size(bytes)	AIFORE	ProFuzzer	AFL-Analyze	TIFF-fuzzer
324	98.91%/91.23%	74.26%/66.67%	55.70%/38.60%	97.40%/(N/A)
198	97.64%/72.23%	50.29%/60.00%	31.96%/12.00%	65.30%/(N/A)
166	84.33%/88.45%	37.01%/40.00%	21.05%/21.67%	82.33%/(N/A)
148	72.23%/85.32%	2.20%/0.00%	1.35%/21.21%	40.00%/(N/A)
114	93.18%/84.23%	73.18%/77.78%	39.66%/44.44%	85.60%/(N/A)
44	100.00%/91.22%	56.25%/63.64%	56.25%/45.45%	100.00%/(N/A)
58	45.34%/84.33%	75.00%/82.35%	28.12%/23.53%	22.22%/(N/A)
5632	94.40%/78.32%	(N/A)/(N/A)*	0.00%/51.02%	33.33%/(N/A)
836	**85.75%/84.42%**	**46.02%/48.81%**	**29.26%/32.24%**	**65.77%/(N/A)**

Evaluation: Format Extraction Time - SOTA comparison

- Average time to parse a file (seconds)
- The average time to parse a file with AIFORE (and TIFF-fuzzer) does not differ significantly for different size classes, while ProFuzzer and AFL-Analyze spend much more time parsing larger files.

Input		Program	Large Seeds					Small Seeds				
			Size(bytes)	AIFORE (B/T)	ProFuzzer	AFL-Analyze	TIFF-fuzzer[2]	Size(bytes)	AIFORE (B/T)	ProFuzzer	AFL-Analyze	TIFF-fuzzer[2]
Trained	ELF	readelf	808	29(24/5)	14,224	14	8	324	28(22/6)	977	6	7
	GIF	gifsicle	695	24(19/5)	84,123	42	91	198	23(17/6)	11,392	5	18
	TIFF	tiffdump	448	43(39/4)	1,529	9	13	166	44(40/4)	145	4	11
	TTF	freetype	542	19(12/7)	2,967	8	13	148	17(11/6)	100	2	6
Untrained	PCAP	tcpdump	894	24(19/5)	26,241	18	8	114	20(14/6)	95	2	8
	WAV	sfinfo	572	21(17/4)	2,993	11	8	44	23(19/4)	27	1	8
	BMP	jasper	630	21(15/6)	2,004	10	6	58	9(3/6)	110	0.1	6
	XLS	xls2csv	6,656	61(56/5)	(N/A)[1]	255	22	5,632	52(47/5)	(N/A)[1]	17	94

Evaluation: Fuzzing Performance - Code Coverage

- The coverage increment with AIFORE is significant, even though the target program and the file format are unseen.
- AIFORE achieves the best performance in most cases.

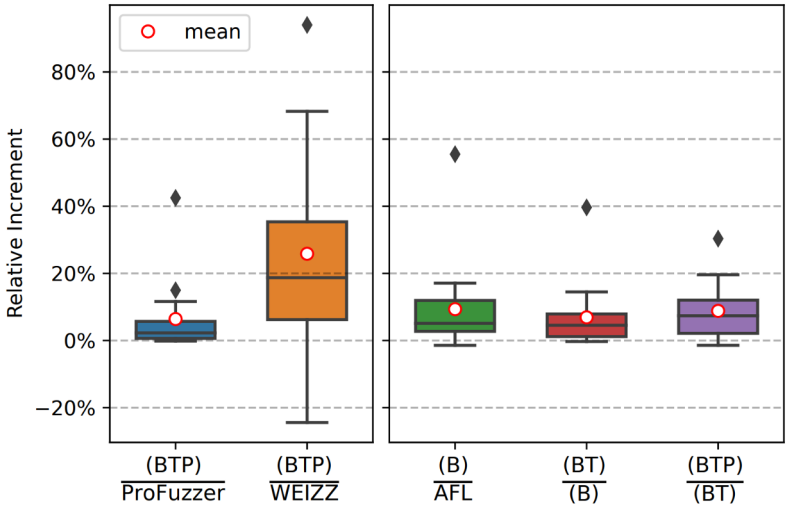
Format	Program[1]	AIFORE (B)[2]	AIFORE (B+T)[2]	AIFORE (B+T+P)[2]	AFL	AFLFast	EcoFuzz	ProFuzzer	TIFF-fuzzer	WEIZZ
BMP	jasper	7123	7705	**7887**	6694	5926	7777	7437	5331	6344
ELF	readelf*	9880	13798	**17985**	9652	12020	13791	17872	2426	15720
	elfutils-readelf	7213	7541	**8308**	6995	5873	6608	7978	2180	6519
GIF	gifsicle*	4702	5062	**5435**	4528	4634	4675	5315	4146	4578
	magick*	14221	14414	**16685**	13529	13684	12335	14512	8351	10002
	gif2tiff*	2458	2451	2576	2372	2466	2500	**2580**	2014	2383
	exiv2	16529	17500	**19245**	14117	14417	16602	18952	7637	11437
JPG	jhead	1228	1258	**1281**	1244	1244	1244	1253	856	1246
PCAP	tcpdump	16772	19201	**22963**	14535	13743	19823	22930	1561	11837
PNG	pngtest	3219	3223	3629	3217	3239	3268	3617	2846	**4802**
TIFF	tiffdump*	1145	1155	**1177**	1054	1073	1104	1135	522	1128
TTF	freetype_parser*	9893	**10520**	10370	6362	6309	10030	7278	4513	7316
WAV	sfinfo	2477	2501	**2632**	2326	2326	2379	2498	2019	2566
XLS	xls2csv	2476	2699	**2706**	2512	2510	2619	2424	1841	2470
ZIP	7za	24696	25266	**28159**	21429	22089	25226	27979	13299	21833

Evaluation: Fuzzing Performance - Bug Detection

- With the help of format knowledge, AIFORE finds 34 bugs (20 are uncovered by other fuzzers) in total after manual deduplication
- Including 10 buffer overflow bugs (CWE-122), 18 NULL pointer dereference bugs (CWE-476), and 6 double-free bugs (CWE-617)
- With 2 new bugs in the newest version of `xls2csv`

Evaluation: Fuzzing Performance - Ablation Study

- Coverage comparison between AIFORE and other SOTA fuzzers, and coverage increment by each module of AIFORE.
- (B) Field boundaries help mutate bytes as a whole in a field, 9.3% increment
- (T) Field types help mutate fields with proper mutators, 6.9% increment
- (P) Power scheduling helps assign more energy to those formats which are not mutated adequately, 8.8% increment



Summary & Conclusion

- Field boundary analysis: taint analysis, BB level merge, minimum cluster
- Field type classification: field feature extraction, CNN classification
- Power scheduling: power rebalance, format-aware mutation, seed filtering

- We implement a novel **smart fuzzing solution AIFORE** and systematically evaluated it on a wide range of input formats and programs. Results show that it has a much better performance on input format recognition and format-aware fuzzing than SOTA solutions.

Q&A
Thank you!