OFFZONE 2025

Поиск иглы в стоге кода: гибридный направленный фаззинг с LibAFL и Sydr

Дарья Парыгина

Старший лаборант ИСП РАН, исследователь по информационной безопасности

@pa_darochek

BIO

- Разработчик-исследователь в ИСП РАН
- Занимаюсь исследованиями в области информационной безопасности и динамическим анализом бинарного кода
- Мейнтейнер символьного интерпретатора Sydr, тулчейна динамического анализа для SSDLC Sydr-Fuzz, анализатора аварийных завершений Casr https://github.com/ispras/casr
- Основатель и мейнтейнер направленного фаззера LibAFL-DiFuzz на основе библиотеки LibAFL
- Разрабатываю фазз-таргеты для open source проектов, в том числе для фреймворков машинного обучения Tensorflow и PyTorch
- Добавлена на доску почёта Google BugHunters за репорт бага в TensorFlow
- Окончила бакалавриат и магистратуру ВМК МГУ
- Соавтор 6 научных статей в сборниках ISPRAS Open, ISPRAS IVMEM, Journal of Mathematical Sciences
- Выступала на научных конференциях IVMEM 2022, ISPRAS Open 2024
- Автор тезисов в журнале «AUTOMATIC CONTROL AND COMPUTER SCIENCES», выступала на конференции МиТСОБИ 2024

OFFZONE
2025

Направленный фаззинг

Зачем нужен направленный фаззинг?

- Фаззинг – динамический подход к автоматизированному тестированию ПО, важная составляющая SSDLC
- Стандартный фаззинг с обратной связью по покрытию испытывает трудности при анализе четко обозначенных участков кода

Направленный фаззинг – вид фаззинга, нацеленный на достижение заданных точек в коде при помощи метрик близости

- Сочетает методологию фаззинга с фокусировкой на заданных областях кода
- Метрика близости позволяет гибко задавать критерий близости к заданным областям
- Применяется для анализа патчей, воспроизведения крешей, верификации предупреждений SAST

Развитие направленного фаззинга

AFLGo

Алгоритм имитации отжига

Задача оптимизации

Цепочки

Целевые последовательности

LOLLY, Berry, LeoFuzz

Оптимизации

Удаление недостижимого кода, адаптивные мутации

BEACON, WindRanger

Runtime

DBI, динамический граф

V-Fuzz, ParmeSan

Интеграция с DSE

DrillerGo, Berry, GTFuzz

ML/DL

V-Fuzz, FuzzGuard, DeFuzz

Проблемы и решения

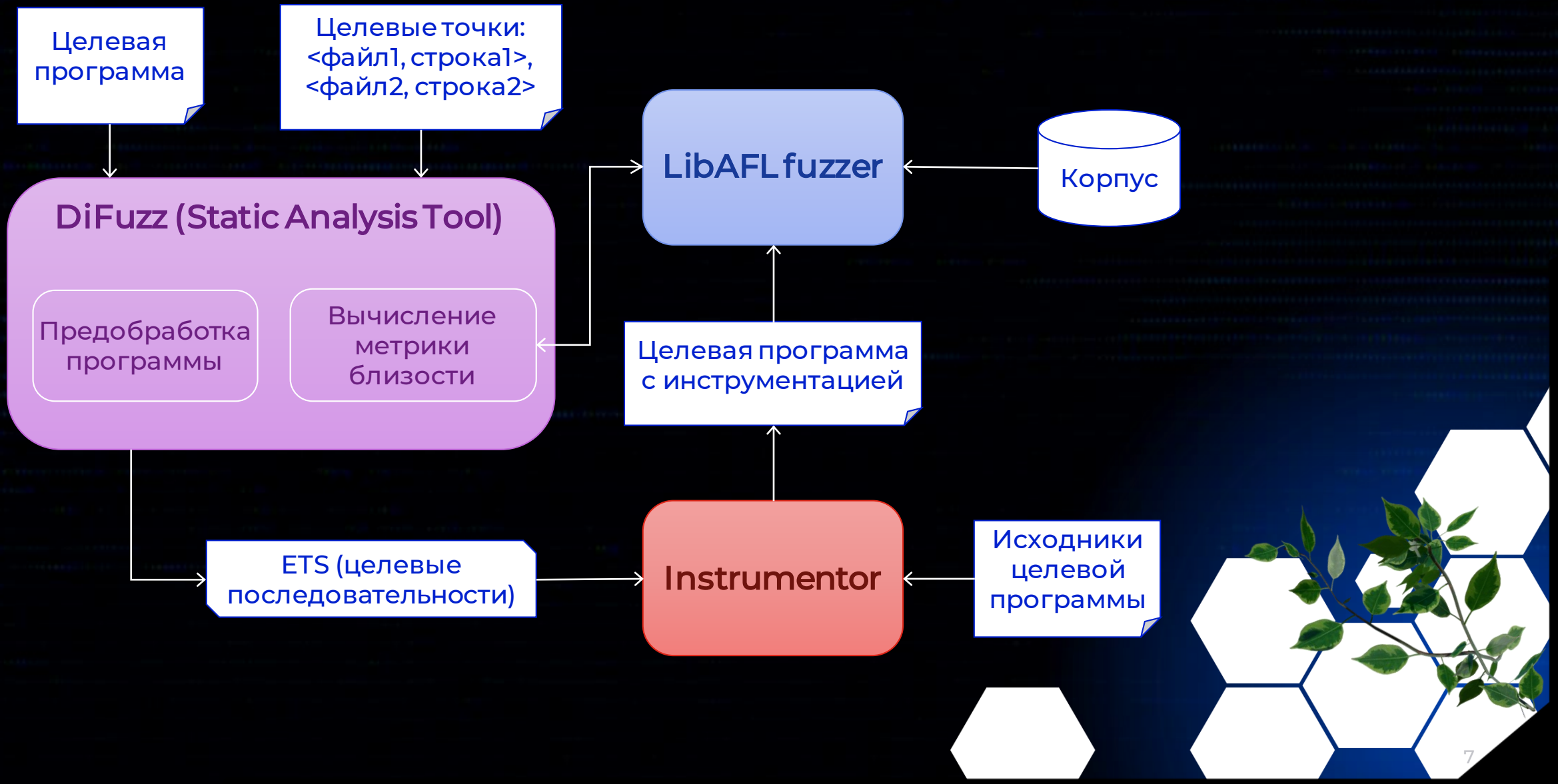
Проблемы

1. Недостаточный анализ структуры программы
2. Игнорирование контекста целевых точек
3. Сложность комбинирования подходов из-за реализаций в небольших независимых прототипах
4. Сложность преодоления сложных условий

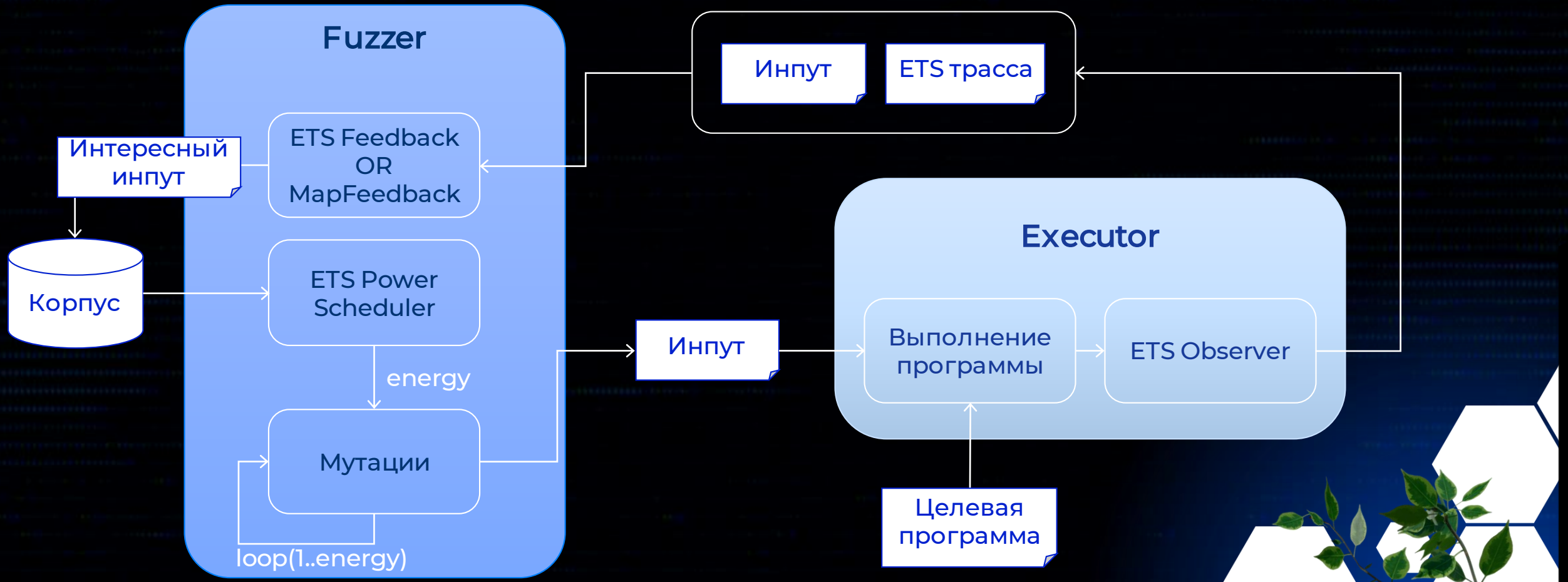
Решения

1. Разрешение неявных вызовов при помощи анализа иерархий наследования
2. Добавление контекстных весов
3. Библиотека LibAFL как базис – модульная архитектура фаззера
4. Интеграция с DSE

Схема LibAFL-DiFuzz

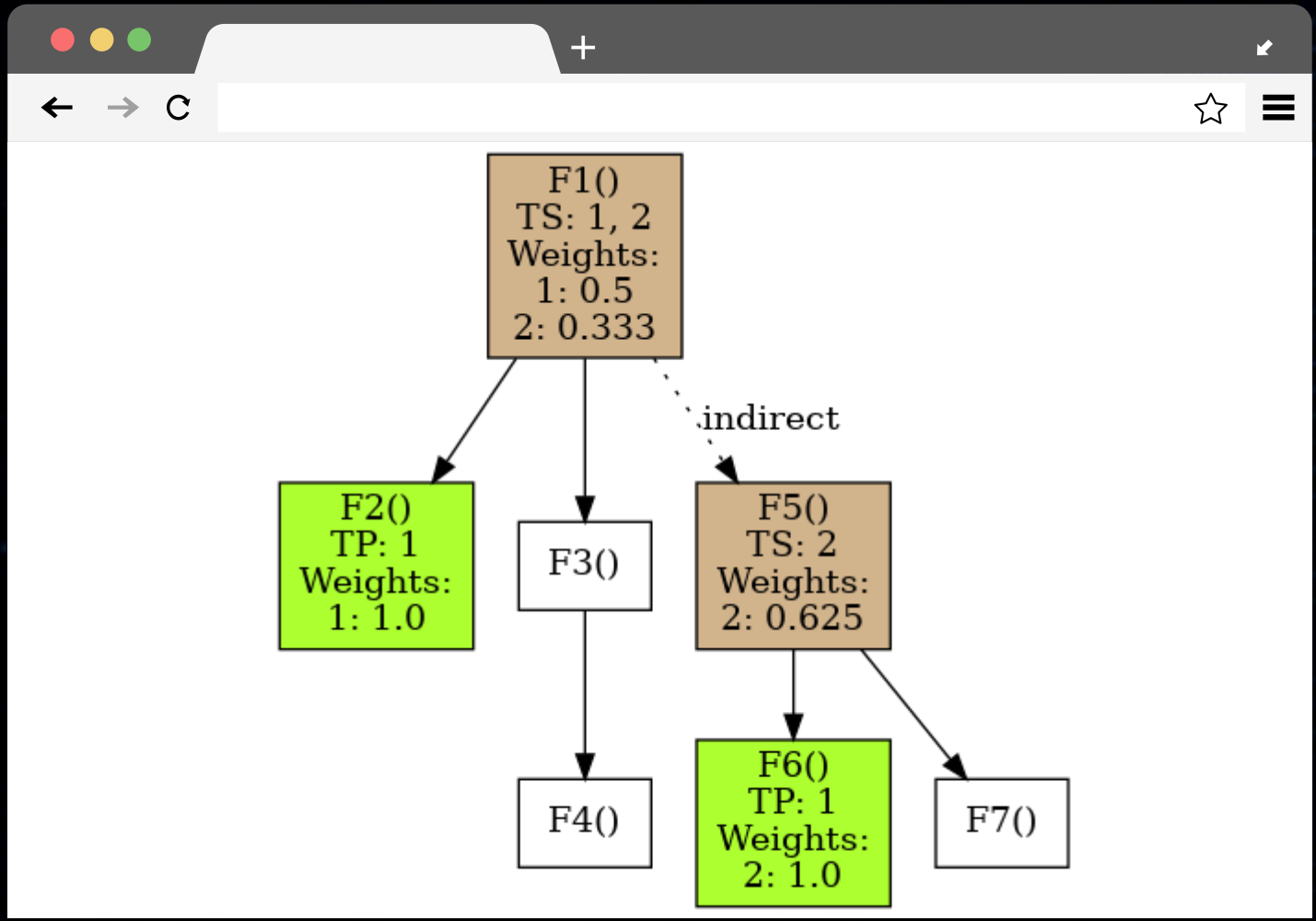


Структура LibAFL фаззера



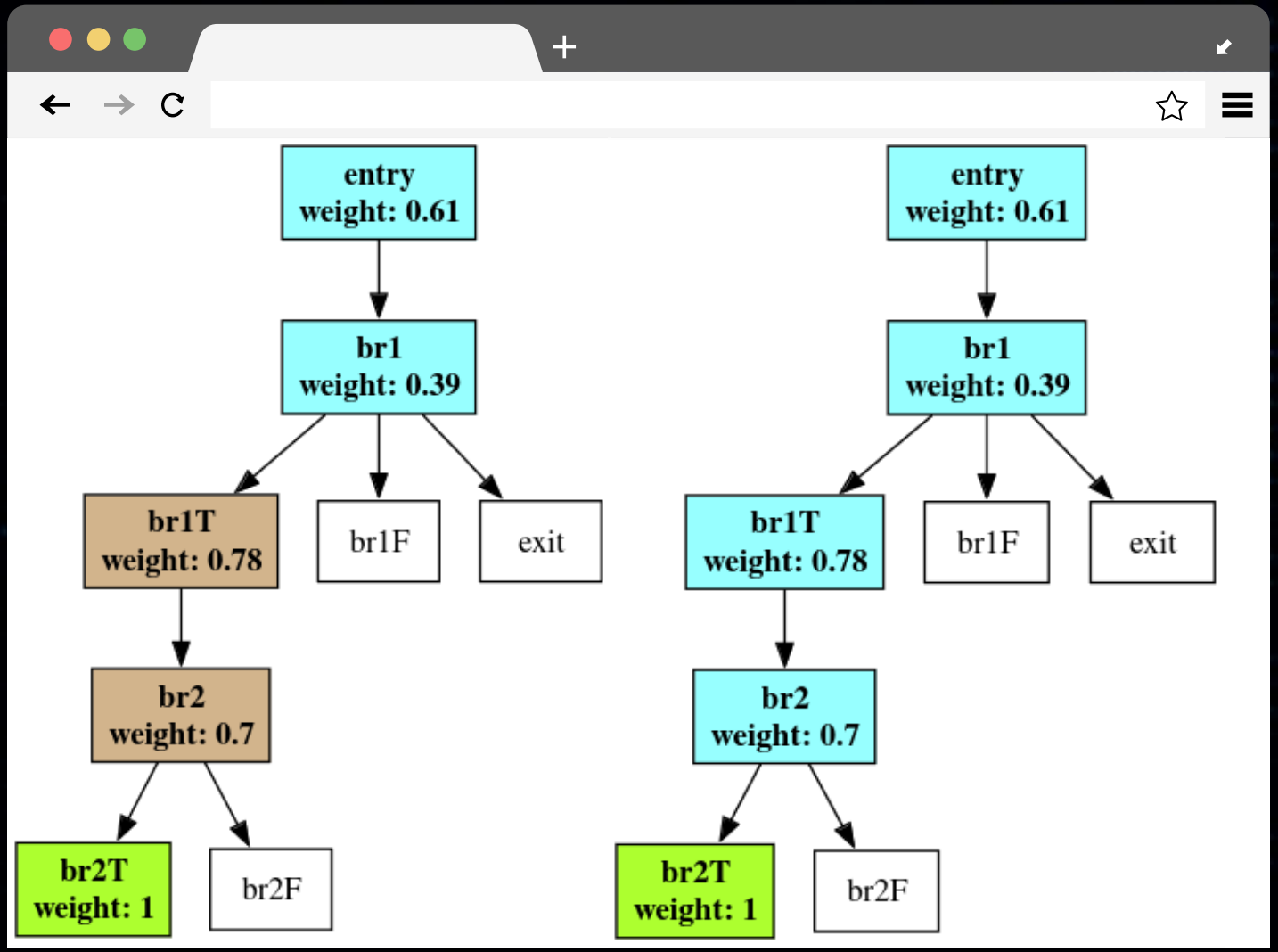
Фичи LibAFL-DiFuzz

- Разрешение неявных вызовов: на основе совпадения иерархии и имени метода
- Целевые последовательности (ETS): доминаторы по функциям и базовым блокам
- Контекстные веса: дистанция до целевой точки и структура графа



Метрика близости

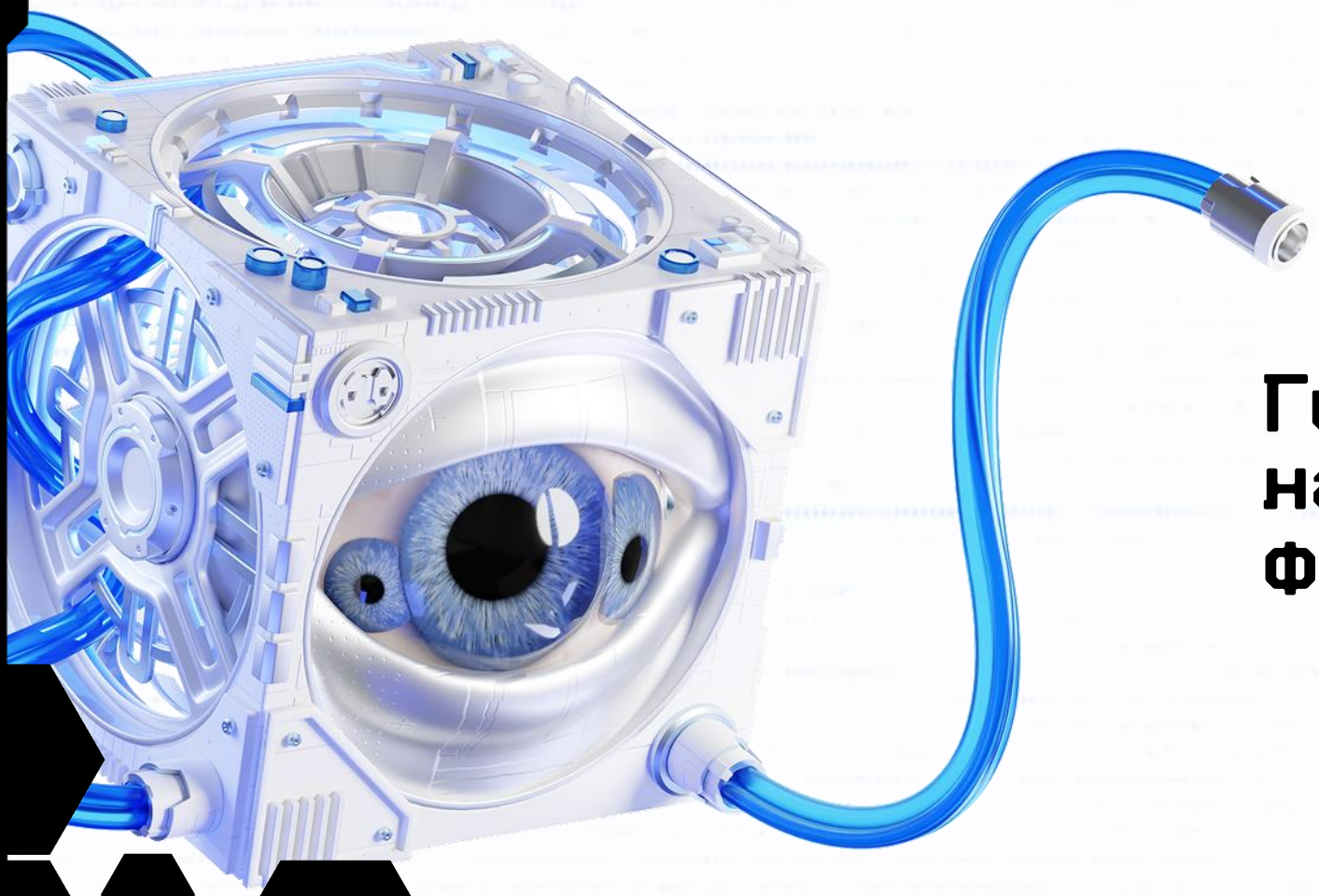
- Строится на основе 3 величин, показывающих схожесть трассы выполнения с ETS
- Чем больше значение, тем ближе трасса к целевой точке
- Используется для вычисления энергии инпута



Оценка LibAFL-DiFuzz

ID	LibAFL-DiFuzz	AFLGo	BEACON
cxxfilt	175	49	60
giflib	3	18	4
jasper	16	16	50
libming	127	412	53
libxml_1	14	28	43
libxml_2	300	343	44

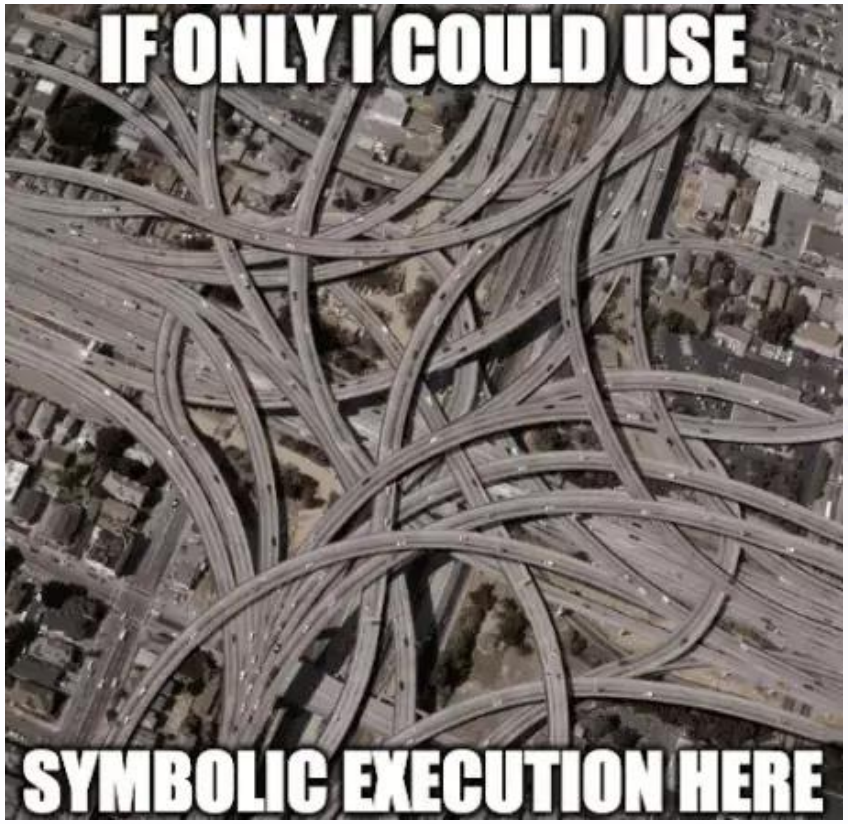
OFFZONE
2025



Гибридный направленный фаззинг

Динамическая символьная интерпретация

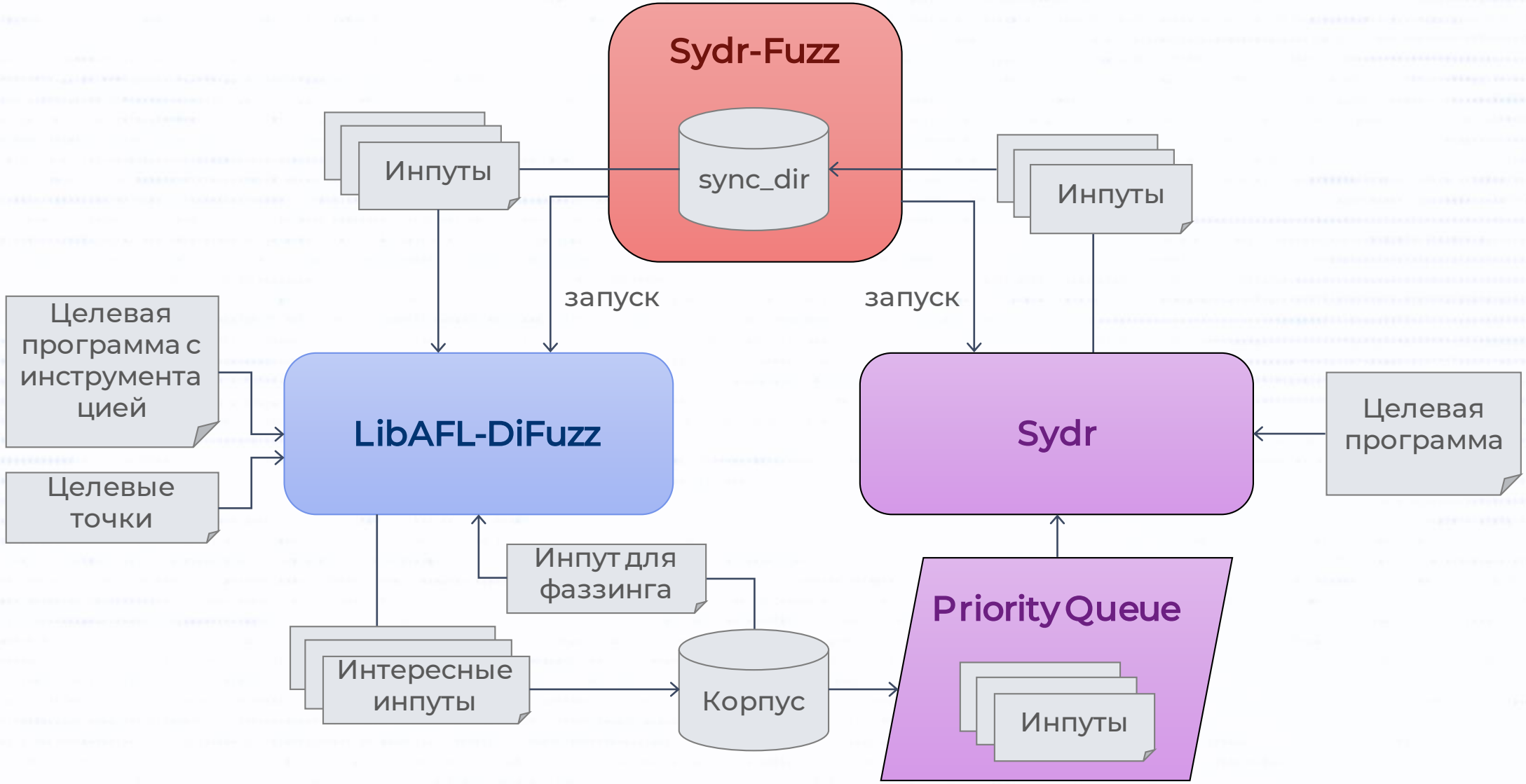
- Сопоставляет входным данным программы специальные символьные переменные
- Строит формулу для каждой инструкции программы в терминах символьных переменных
- Составляет предикат пути – конъюнкцию всех условий переходов
- Подбирает значения входных данных, используя SMT-решатель
- Очень медленная сама по себе



[https://www.code-intelligence.com/blog/using-symbolic-execution-fuzzing]

Направленный фаззинг + DSE
=
скорость + преодоление
сложных ограничений

Схема гибридного направленного фаззинга



Пайплайн Sydr-Fuzz

Sydr-воркер

- Вход: программа + инпут из вершины Priority Queue
- Выход: инпуты, открывающие новые пути вдоль трассы выполнения

Sydr-Fuzz

- Обновляет Sydr Priority Queue из корпуса фаззера
- Запускает Sydr-воркеры
- Экспортирует новые инпуты от Sydr в sync_dir
- Печатает статистику
- Минимизирует и сортирует результаты

LibAFL-DiFuzz

- Мутирует инпуты из корпуса
- Вычисляет полезность и метрику близости для инпутов
- Добавляет новые полезные инпуты в корпус
- Импортирует инпуты от Sydr из sync_dir

Sydr Priority Queue

- Нужна для упорядочивания инпутов, приходящих от фаззера в Sydr
- Основана на бинарной куче
- Новые инпуты от фаззера добавляются в отсортированном порядке
- Верхний элемент имеет самый высокий приоритет

Метрики для приоритета

- **ETS score**: есть ли в трассе выполнения хотя бы 1 элемент из ETS
- **Coverage score**: открывает ли новое покрытие
- **File score**: время создания файла / размер файла

Импорт инпутов в LibAFL-DiFuzz

- Для фаззинга запускается несколько процессов-воркеров
- Sync-процессы аналогичны процессам-воркерам, но содержат дополнительный этап (импорт инпутов)
- Все процессы LibAFL общаются друг с другом при помощи сообщений
- При импорте инпута в sync-процессе он отправляет broadcast-сообщение остальным процессам
- Процессы-воркеры обрабатывают сообщение и добавляют инпут в свой корпус
- Sync-процессы также обрабатывают сообщения о новых инпутах от процессов-воркеров, поддерживая своё состояние актуальным
- Использование отдельных sync-процессов позволяет оптимизировать процесс импорта инпутов, т.к. во время импорта фаззинг останавливается (архитектурная особенность LibAFL)

Минимизация и сортировка результатов

- **Objective** – инпут, который приводит к одному из событий: креш / зависание / достижение целевой точки
- **Минимизация**: по уникальности ETS трассы (трасса выполнения, из которой удалены элементы, не входящие ни в одну ETS)
- **Сортировка**: по достигнутым целевым точкам + отдельно креши

Objective 1	Objective 3	Objective 4
ETS трасса: [bb1, **bb2**, bb5]	ETS трасса: [bb1, **bb2**, bb3, bb5], CRASH	ETS трасса: [bb1, **bb4**, bb5]

Objective 2	Objective 5
ETS трасса: [bb1, bb2, bb5]	ETS трасса: [], CRASH

Кластер 1 Целевая точка: bb2	Кластер 2 Целевая точка: bb4	Кластер 3 CRASH
Objective 1 Objective 3	Objective 4	Objective 3 Objective 5

Оценка LibAFL-DiFuzz + Sydr

ID	Sydr-Fuzz	LibAFL-DiFuzz	AFLGo	BEACON	WAFLGo	Wind Ranger	FishFuzz	Prospector
cxxfilt_1	17	175	49	60	41	44	384	470
cxxfilt_2	219	-	63	55	3043	704	1208	347
giflib	3	3	18	4	5	55	41	45
jasper	18	16	16	50	-	81	287	22
libming	32	127	412	53	-	-	862	59
libxml_1	4	14	28	43	107	666	2285	1694
libxml_2	28	300	343	44	-	1108	-	-

Итоги

- LibAFL-DiFuzz – направленный фаззер с гибкой конфигурируемой архитектурой на основе библиотеки LibAFL
- Разрешение неявных вызовов, метрика близости на основе ETS, контекстные веса повышают эффективность направленного фаззинга
- Реализация в пайплайне Sydr-Fuzz для организации взаимодействия фаззера с Sydr
- Импорт инпутов в фаззер в отдельных процессах для снижения временных расходов
- Минимизация и сортировка инпутов для упрощения анализа результатов
- Гибридный направленный фаззинг увеличивает эффективность анализа

#include <offzone>

2025 OFF + ZONE