НЕ ВСЕ ДЕФЕКТЫ ОДИНАКОВЫ:
КАК СТРУКТУРА SAST-НАХОДКИ ЛОМАЕТ LLM-ТРИАЖ

ZeroFalse: property-сработки и trace-сработки доказываются по-разному —
и один универсальный вопрос к модели хоронит 9 из 10 реальных уязвимостей

Юрий Туманов (Эльария) · AppSec · «Ростелеком» · исследование 02–09.2026

ISCRA TALKS 2026

тезаурус: SAST — сканер, читающий исходный код без запуска и кричащий на подозрительное · сработка (находка) — его сигнал «тут может быть дыра» · триаж — разбор: реальна ли проблема · LLM — большая языковая модель · property / trace — два типа сработок, о них весь доклад

ОБ АВТОРЕ

КТО

Юра Туманов
ака Эльария / Psycho Drake
«Ростелеком» · AppSec · MLSecOps

ЭКСПЕРТИЗА

кибербез с 2001 года
кандидат технических наук @ НИЯУ «МИФИ»
кибербез + нейронки · пою

КОНТАКТЫ

eljees.github.io

@psychodrake · github.com/Eljees
3.14hell@gmail.com

ПОЧЕМУ ЭТА ТЕМА

Покупалась RTX 4080,
чтобы играть в киберпанк.
Служит — чтобы создавать
его в реальной жизни.

тезаурус: AppSec — безопасность приложений · MLSecOps — применение машинного обучения в задачах безопасности · RTX 4080 — игровая видеокарта, на которой крутятся все модели доклада

ДЕСЯТКИ ТЫСЯЧ ФОЛЗОВ В НЕДЕЛЮ

масштаб ручного разбора сработок SAST у нас в компании — оценки сверху

ДЕСЯТКИ ТЫСЯЧ

сработок SAST приходит в пиковые недели — это входящая очередь на разбор

95 %

из них — фолзы: сканер кричит, а уязвимости нет. Но узнаёшь это, только разобрав сработку

< 10

человек в AppSec-команде на весь этот поток

ЧТО ЭТО ЗА ПОТОК

70+ приложений · ~3 000 репозиториев · 150+ языков

в агрегаторе накоплено > 1 000 000 сработок

93 % из них — property-сработки (пересчитано поштучно, см. дальше)

тысячи сработок на человека в неделю — меньше минуты на решение

Дальше в докладе — кому и по каким правилам отдать эту рутину, не теряя реальные уязвимости. И почему ответ зависит от того, как устроена сама сработка.

тезаурус: SAST — робот, читающий исходный код без запуска и кричащий на подозрительное · фолз — ложная тревога сканера · триаж — разбор: реальна ли проблема · агрегатор — система, где копятся сработки всех сканеров

ПЛАН ДОКЛАДА

шесть остановок — от наивной гипотезы к анатомии сработки и живым цифрам

01 Складно ≠ доказано: почему «отдай сработку LLM — пусть решит» не работает, и матмодель с правом на Unknown

02 Два типа сработок — property-based и trace-based — и эксперимент, где один чужой вопрос похоронил 140 из 363 реальных

03 Архитектура ZeroFalse: роутер по структуре доказательства, модель предполагает — улики подтверждают — правила решают

04 Property-линия: семь подтипов P0–P6, пять этапов проверки, пять семейств на 242 … 1 007 эталонах

05 Trace-линия: как подать контекст, evidence-gate «перечисли факты до вердикта», инъекции на живом приложении

06 Цифры: экзамен против 2 546 решений людей · грабли · что унести с собой

тезаурус: Unknown — третий ответ системы «не знаю, пусть смотрит человек» · роутер — маршрутизатор, отправляющий сработку в нужную ветку разбора · evidence-gate — форма допроса модели «сначала факты, потом вердикт» · эталон — сработка, у которой уже есть вердикт человека; на эталонах машину проверяют · P0–P6 — семь подтипов property-сработок; каждый термин раскрыт на своём слайде

СКЛАДНО ≠ ДОКАЗАНО

исходная гипотеза: один универсальный промпт, большая LLM сама разберёт, что фолз, а что нет. Проверка: 100 сработок SQL-инъекций (PHP) через корпоративный LLM-портал

РЕАЛЬНАЯ ВЫДАЧА, ДОСЛОВНО

```
вердикт: TP · confidence 1.00
ссылка: строка 32358 · в выданном коде: 307 строк
«htmlspecialchars() нейтрализует SQL-инъекцию» · conf 0.9
```

(htmlspecialchars экранирует HTML-символы — от SQL-инъекции она не защищает)

10 / 100 вердиктов ссылаются на строку кода, которой НЕТ в выданном фрагменте — модель «увидела» несуществующий код

9 / 10 из этих десяти закрывают сработку как реальную уязвимость с уверенностью 1.00

50 % случаев модель противоречит сама себе, когда тот же фрагмент показывают повторно (17 из 34)

Текст безупречен — вердикт выдуман. Вопрос всего доклада: как не дать убедительности сойти за доказательство.

LLM-портал — корпоративный веб-доступ к большой модели Qwen2.5-72B: формат ответа не зафиксировать (ни JSON-схемы, ни guided decoding). Дальше в докладе — только локальные модели под полным контролем

тезаурус: промпт — текст задания модели · TP (true positive) — вердикт «реальная уязвимость» · confidence — уверенность, которую модель сама заявляет в ответе · JSON-схема / guided decoding — способы зафиксировать формат ответа модели на стороне сервера

МАТМОДЕЛЬ: ПРАВО НА UNKNOWN

стоим на плечах титанов — классическое распознавание образов конца 1950-х

РАСПОЗНАВАНИЕ ОБРАЗОВ

сработка с уликами = объект с признаками; раскладываем на классы «фолз» и «реальная»

цены ошибок несимметричны: лишний фолз человеку — минуты; похороненная реальная — дыра в проде

ПРАВО НА ОТКАЗ — ПРАВИЛО ЧОУ

классика 1957/1970 (reject option): если ошибки дорогие — разреши третий ответ, Unknown: «пусть смотрит человек». Суммарно это дешевле, чем заставлять угадывать

покрытие = доля сработок с ответом; риск = доля ошибок среди закрытых

ФАКТОРИЗАЦИЯ ЗАДАЧИ

вместо одного «разбери всё» — много маленьких задач:

(структура доказательства) × (класс дефекта)

у каждой — свой инструмент, свой замер и свой порог. Первый множитель — тема доклада

«похоронить» = закрыть реальную уязвимость как фолз: сработка исчезает из очереди навсегда, дыра остаётся в проде. Эта цифра дальше — главная.

тезаурус: классификатор — программа, относящая объект к одному из классов · reject option (правило Чоу) — право классификатора ответить «не знаю» · класс дефекта — тип уязвимости по классификатору CWE · факторизация — разложение одной большой задачи на независимые маленькие

ДВА ТИПА СРАБОТОК: PROPERTY И TRACE

у сработки есть структура доказательства — и обрабатывать два типа надо по-разному

PROPERTY-BASED · «УЯЗВИМОСТЬ-СВОЙСТВО»

опасен сам ФАКТ в коде:

секрет или пароль в коде (CWE-798 · CWE-257 · CWE-259)

передача без шифрования (CWE-319) · слабый хеш или случайность (CWE-328 · CWE-330)

слишком открытые права (CWE-276 · CWE-732) · пустой catch — проглоченная ошибка (CWE-396 · CWE-397)

устаревший API (CWE-477) · чувствительное значение в логе или ответе (CWE-359 · CWE-209)

ключевой вопрос: настоящий пароль или заглушка? боевой конфиг или тест?

TRACE-BASED · «УЯЗВИМОСТЬ-ПУТЬ»

опасен МАРШРУТ данных: source › sink

SQL-инъекция (CWE-89) · XSS (CWE-79)

инъекция кода (CWE-94) · инъекция команд (CWE-78)

обход каталогов (CWE-22) · SSRF (CWE-918) · десериализация (CWE-502)

ключевой вопрос: дошли ли данные атакующего до опасного вызова — и была ли по дороге защита?

словарик: source — место, куда атакующий может подать свои данные · sink — вызов, где эти данные становятся опасными (запрос к БД, вывод в HTML, команда ОС) · CWE — международный классификатор типов уязвимостей: номер в скобках = тип дефекта · агрегатор — система, где копятся сработки всех сканеров

93 % потока агрегатора — property-based (1 004 440 сработок, разложены по типу поштучно; в замере на 2 546 — 74 %). Большинство сработок вообще не про «путь атаки» — и всё же у модели спрашивают именно про него

СПРОСИ НЕ ТАК — ПОТЕРЯЕШЬ 9 ИЗ 10

взяли 38 классов дефектов — по 10 реальных и 10 фолзов на каждый — и задали всем сработкам один и тот же вопрос: «покажи путь атаки»

TRACE-КЛАССЫ — ПУТЬ ЕСТЬ
похоронено реальных: 0

вопрос подошёл: у пути есть трасса, её можно проверить; фолзы отсеяны верно

PROPERTY — ПУТИ НЕТ

похоронено: 7–9 из 10

модель ищет путь, не находит — и решает «значит, не уязвимость». А путь здесь и не должен существовать

140 / 363

реальных уязвимостей похоронил один неправильный вопрос. Хуже всего пришлось самым простым и массовым классам — паролям и секретам в коде

Отсюда тезис доклада: сначала определить, как в принципе должно выглядеть доказательство для этого класса — и только потом спрашивать модель, есть ли уязвимость.

тезаурус: «похоронить» — закрыть реальную уязвимость как фолз · класс дефекта — тип уязвимости по CWE · промпт — текст задания модели; здесь один на все классы · Qwen2.5-Coder-14B — локальная модель на 14 миллиардов параметров

АРХИТЕКТУРА ZEROFALSE

принцип: модель предполагает — улики подтверждают — правила проверяют — последнее слово всегда за системой правил · название — оммаж ZeroFalse (arXiv:2510.02534, 2025)

РОУТЕР

по структуре доказательства: правило сканера + сигнатура; CWE — подсказка, не ключ

→ **property-based** › пять этапов: правила до модели → анкета с цитатами → постпроверки → gate (слайды 10–11)

→ **trace-based** › пакет улик по трассе + evidence-gate «перечисли факты до вердикта» (слайды 14–16)

МНЕНИЕ МОДЕЛИ

вердикт · обоснование · confidence — само по себе не решает ничего

→

УЛИКИ

цитаты из кода · трасса · строки происхождения данных · факты движка

→

ПРАВИЛА

fastpath без LLM · 120+ детерминированных проверок · пороги по классам · канарейки

→

РЕШЕНИЕ СИСТЕМЫ

закрыть как фолз · завести дефект · отдать человеку (Unknown)

СКВОЗНОЙ ЖУРНАЛ: каждое решение воспроизводимо из лога прогона — все цифры этого доклада посчитаны из него. Модель никогда не хоронит сработку единолично.

как читать цвета: чёрное — мнение (не решает) · тёмно-красное — проверяемые факты и алгоритмика · оранжевое — единственное, что уходит наружу

тезаурус: роутер — маршрутизатор: определяет тип сработки и отправляет в нужную ветку · сигнатура доказательства — набор признаков правила сканера: нужен ли путь данных, нужна ли роль значения… (слайд 10) · fastpath — решение правилами до вызова модели · детерминированный — всегда одинаковый результат, без модели · канарейка — подмешанная сработка с заведомо реальным дефектом

АНАТОМИЯ PROPERTY-СРАБОТКИ: 7 ПОДТИПОВ

у 180 property-классов общее одно: доказательство не требует пути source › sink. Внутри — семь подтипов (чем нарушено правило) с разной глубиной улик (сколько надо знать о программе); они объясняют ошибки триажа лучше, чем размер модели или номер CWE

	подтип	что нарушено	пример	глубина улик	что может модель (роль LLM)
P0	**Значение**	само значение нарушает политику	`password = "admin123"` · `chmod 0777` · токен в литерале	D0 байты	не нужна: хватает регулярки и энтропии
P1	**Символ**	вызван запрещённый / устаревший API	`gets()` · deprecated-перегрузка (CWE-477)	D1 символ	найти вызовы через функции-посредники (обёртки)
P2	**Отношение**	неверна комбинация ≥ 2 свойств одного места	`target="_blank"` + внешний href – `noopener` (CWE-1022)	D2 связка	подсказать значения вычисляемых атрибутов; программа перепроверяет
P3	**Состояние**	нарушен порядок операций над объектом	`encrypt()` до `init()` · use после close	D3 порядок	вытащить из документации порядок вызовов; проверяет человек
P4	**Обязанность**	обязательного действия нет хотя бы на одном пути	пустой `catch` · сбой аутентификации без лога (CWE-778)	D3–D4	назвать, какая операция могла упасть; отсутствие обработчика доказать не может
P5	**Роль**	слабый примитив используется в security-роли	`java.util.Random` → токен сброса пароля (CWE-338) · ошибка → клиенту (CWE-209)	D4 роль	главная роль: понять, для чего используется значение, и показать строку-цитату
P6	**Окружение**	истина зависит от внешнего состояния	жив ли токен · откуда загрузится библиотека (CWE-426)	D5 вне кода	лишь выбрать, какую внешнюю проверку запустить

глубина улик — минимум знания о программе, без которого вердикт невозможен: **D0** сами байты значения · **D1** разобранное имя функции · **D2** связка свойств одного места · **D3** порядок выполнения строк · **D4** роль значения в программе · **D5** факт вне исходного кода; чем глубже — тем нужнее модель и дороже проверка её ответа

CWE говорит, о чём сработка; сигнатура доказательства — как её доказывать.

NOT FOUND ≠ DISPROVED: «не нашёл» не равно «опроверг». Для P4 и P6 много Unknown — норма.

тезаурус: литерал — значение, вписанное в код прямо текстом или числом · символ — имя, которое разбор кода привязал к конкретной функции · API — функции библиотеки, которые вызывает программа; deprecated — устаревшая, производитель просит не использовать · регулярка — шаблон поиска по тексту · энтропия — мера случайности строки · обёртка — своя функция-посредник вокруг чужой · сигнатура доказательства — набор из 11 признаков правила сканера: как его доказывать · Unknown — ответ «не знаю, пусть смотрит человек»

PROPERTY: ПЯТЬ ЭТАПОВ ВМЕСТО ОДНОГО ВОПРОСА

у «факта в коде» всегда есть ступень проверки, не зависящая от настроения нейронки; модель находит — программа проверяет

1 **ПРАВИЛА ДО МОДЕЛИ ·** путь файла и синтаксис строки: vendor/, SBOM, xmlns=, минифицированный бандл, закомментированная строка. Решает от ¼ до ⅔ закрытий — без LLM

2 **КАРТОЧКА И ЗАДАНИЕ ·** строка + код вокруг + «факты движка»: схема, хост, класс хоста, найденный клиентский вызов, флаги ценности канала

3 **МОДЕЛЬ ЗАПОЛНЯЕТ АНКЕТУ ·** узкие фактические вопросы, JSON под guided decoding; на КАЖДОЕ утверждение — дословная цитата. Поле «вердикт» есть, но gate им не пользуется

4 **ПОСТПРОВЕРКИ ·** программа проверяет ответ модели: каждую цитату ищет в карточке буква в букву — не нашлась, утверждение выбрасывается; ответ сверяет с фактами движка (модель говорит «адрес локальный», разборщик видит внешний домен — противоречие, к человеку); уверенность считает по числу подтверждённых цитат: ни одной — не выше 0,55, одна — не выше 0,75, закрыть можно только от 0,85 — одной цитаты для закрытия не хватает никогда

5 **GATE ПО ПИСЬМЕННОЙ ПОЛИТИКЕ ·** закрыто автоматом · человеку с причиной · не решено. Предохранители: канарейки в потоке, рубильник (одна задавленная канарейка выключает класс), 3 % ручного аудита

ЗАЧЕМ ВООБЩЕ МОДЕЛЬ

программа умеет проверять, но не умеет находить. Замер: 104 размеченные сработки, одна модель:

только правила: ~½ закрытий, 0 канареек

верим вердикту модели: 38 % закрыто, задавлено 4 из 29 канареек

правила + анкета + gate: 67 % закрыто, 0 из 29

нужны обе половины

ПРАВИЛО: для property-сработки «не вижу пути атаки» — запрещённая причина закрытия; такое закрытие отменяется gate

тезаурус: пак — текст задания модели (промпт) · анкета, JSON — ответ в виде структуры «поле — значение» · guided decoding — сервер модели физически не выпускает ответ вне схемы (5 366 обращений, 0 испорченных) · факты движка — то, что программа-разборщик выяснила сама · gate — ограничитель: выносит решение по письменной политике класса · канарейка — подмешанная сработка с заведомо реальным дефектом · рубильник — одна задавленная канарейка выключает автозакрытие класса

ПЯТЬ СЕМЕЙСТВ PROPERTY-ЛИНИИ

семейство = группа правил сканеров с одной природой сработки; у каждого — план методики, эталоны (сработки с вердиктами людей, которых машина не видела) и канарейки в каждом прогоне · 08–18.09.2026

общий счёт: на эталонах 0 похороненных дефектов в exc, secrets и sensitive; 16 ошибочных закрытий http — 10 из них на 131 сработке моей собственной разметки (она строже политики); канареек задавлено 0 из 247

CWE-477 устаревшие функции (API) **P1**

209 781 в очереди (22 % property-потока) → 96 % — шум из чужих библиотек в репозитории (vendored); закрывается правилом пути файла, без модели

exc — необработанное исключение CWE-248/390/396 **P4**

242 эталона (213 ложных / 29 дефектов): 113 закрыто, 0 ошибок, полнота 0,531; живая очередь — закрыто 74 %, из них 94 % по пути файла; канарейки 60/60

http — адрес http:// вместо https:// CWE-319 **P5**

1 007 эталонов (764 ложных / 243 дефекта): 417 закрыто, точность 0,962; 40,5 % закрытий без модели (точность 0,994); повтор — те же решения 1 107/1 107; канарейки 100/100

secrets — секрет в коде CWE-798 **P0/P6**

999 эталонов (700 ложных / 299 дефектов): 78 закрыто, 0 расхождений с человеком; канарейки 32/32. Прежний конвейер (старый триаж одним промптом) закрыл 281 — среди них 35 настоящих секретов

sensitive — данные не на месте · CWE-359/244/209/499/201/598 **P5**

489 эталонов (все ложные): прежний конвейер закрыл 160 и 122 назвал «дефектом» ошибочно; property — 217 и 0. Три живых батча × 1 000; в агрегатор записано 848 закрытий; канарейки 55/55

Полнота — свойство кода: где много чужих библиотек и документации, закрывается половина очереди; в старом процедурном PHP, где каждая находка — живой вызов, закрывать нечего (http: 52 % → 34 % между приложениями при той же точности)

тезаурус: эталон — сработка, у которой уже есть вердикт человека из истории агрегатора; машина вердиктов не видит, её решения сверяют с ними — только на эталонах считаются точность (доля правды среди закрытых машиной) и полнота (доля ложных сработок эталона, снятых с человека) · канарейка — подложная сработка с заведомо реальным дефектом, подмешанная в прогон · живая очередь — сработки со статусом «To verify», ещё никем не разобранные · батч — партия из 1 000 сработок живой очереди

ЧЕМУ НАУЧИЛИ ПЯТЬ СЕМЕЙСТВ

пять уроков о том, где на самом деле теряется и набирается качество триажа; каждый — из разбора конкретного расхождения машины с людьми на эталонах (сработках с вердиктами людей)

1	**САМОЕ МАССОВОЕ ЗАКРЫВАЕТСЯ БЕЗ МОДЕЛИ — ПО ПУТИ ФАЙЛА**	Путь файла и синтаксис строки закрывают от ¼ до ⅔ сработок семейства. Пример: у одного правила сканера («чувствительные данные в коде», Java) 17 707 из 17 766 сработок — строки файла bom.xml, который сборка генерирует сама (перечень зависимостей, SBOM). Показывать их модели незачем
2	**ПОД ОДНИМ ИМЕНЕМ СЕМЕЙСТВА — ДВА РАЗНЫХ СОРТА ПРАВИЛ**	В семействе «секреты» одни правила ловят код, который работает с учётными данными (переменную с именем password передали в функцию) — за всю историю агрегатора 0 подтверждённых дефектов на 59 тысяч сработок; другие ловят сам секрет — строку, похожую на ключ, — там дефектов две трети. Одно правило gate (ограничителя, который выносит решение по письменной политике) для первого сорта подняло долю закрытий с 3 % до 22 % при нуле ошибок
3	**ЧУЖОЙ ВОПРОС ДАЁТ ЧУЖИЕ ОШИБКИ**	Прежний конвейер (старый триаж одним промптом про путь данных) закрывал исключения с формулировкой «не вижу недоверенного входа, достигающего опасного вызова» — 90 из 320 закрытий; для обязанности (подтип P4: обработчика нет) путь данных ни при чём. Он же закрыл 35 живых ключей как заглушки, потому что файл лежал в папке test/, а в семействе sensitive назвал «дефектом» 122 из 489 ложных сработок — каждую четвёртую
4	**ПОЛНОТУ ПОДНИМАЮТ ФАКТЫ О ПРИЛОЖЕНИИ, НЕ РАЗМЕР МОДЕЛИ**	Полнота — доля ложных сработок, снятых с человека. В семействе исключений 69 из 100 нерешённых сработок закрыл бы один проверенный человеком факт «в этом приложении есть глобальный обработчик ошибок» (rescue_from в Rails, @ControllerAdvice в Spring): один факт = 230 сработок. Обратный пример — http: поднять полноту до 78 % можно только правилом, которое даёт 99 ошибочных закрытий вместо 16; правило выключено
5	**КТО РАЗМЕЧАЛ ЭТАЛОНЫ — ВАЖНО**	В семействе http на 876 сработках, размеченных другими сотрудниками, точность машины 0,985 (6 ошибок на 406 закрытий); на 131 сработке моей разметки — 0,091 (10 ошибок на 11). Моя разметка строже письменной политики семейства: я считал дефектом конфиги с адресом http://127.0.0.1, политика их закрывает. Эталоны надо чистить так же строго, как модель

тезаурус: точность — доля правды среди закрытых машиной · vendored — чужие библиотеки, лежащие в репозитории · факт о приложении — проверенное человеком утверждение о приложении целиком, которого в коде одной сработки не видно

TRACE: КАК ПОДАТЬ МОДЕЛИ КОНТЕКСТ

живой замер: 100 сработок (50 реальных + 50 фолзов), одна модель 14B — меняется только способ подать код; сравнение с боевыми вердиктами ревьюеров

способ подать код	покрытие	верно	похор.	цена
full file — файл целиком	36 %	92 %	3	~7 600 токенов
distilled — только строки трассы	39 %	92 %	3	~3 400 (−55 %)
no-Unknown — «решай всё»	95 %	56 %	**40**	минимум контекста
staged — пошаговый разбор	79 %	73 %	17	3+ вызова модели
cascade — distilled, не уверен › staged	84 %	77 %	16	~3 900 · баланс

экстремум: сработка из обфусцированного кода тянула 418 000 токенов › после дистилляции 1 200

Полный вход ≠ лучший вход. Решительность покупается опасными ошибками — 40 похороненных у «решай всё». Каскад — рабочий баланс; его 16 опасных дальше добивает evidence-gate (следующий слайд). Для trace улика — это восстановленный путь данных; просто «больше кода» уликой не становится.

тезаурус: трасса — цепочка строк кода, по которым данные идут от source (вход) к sink (опасный вызов) · токен ≈ 3–4 символа кода — единица, которой модель читает текст · контекстное окно — сколько токенов модель видит за раз · покрытие — доля сработок с определённым ответом · похоронено — реальные, закрытые как фолз · evidence-gate — «сначала факты, потом вердикт» (следующий слайд)

EVIDENCE-GATE: ФАКТЫ ДО ВЕРДИКТА

модель не имеет права сразу на вердикт — сначала обязана перечислить факты; вердикт вычисляется из таблицы фактов механически (пример — SQL-инъекции)

ШАГ 1 найди настоящий sink — строку, где реально выполняется опасный вызов. Сканер её часто теряет: у нас 0 из 200 сработок доводили sink сами

ШАГ 2 перечисли КАЖДУЮ переменную запроса: откуда пришла · какой строкой · была ли по дороге защита · в каком месте запроса стоит (экранирование действительно только для значения в кавычках)

ШАГ 3 выведи список незащищённых «грязных» переменных

ШАГ 4 вердикт МЕХАНИЧЕСКИ из списка: не пуст › Confirmed · пуст и происхождение всех известно › фолз · улики повреждены › всегда Unknown. «Не переопределяй интуицией»

63 % › 100 %

точность подтверждений на контрольном наборе, проверенном вручную: 19 сработок (12 инъецируемых + 7 безопасных)

что дало прирост: обязательная таблица фактов + доставка строк происхождения кода — модель перестала «вспоминать» код, которого нет

guided decoding: поля ответа обязательны на уровне формата — пропустить шаг физически нельзя

отвергнутая идея: перепроверка второй моделью «докажи обратное» — модель, которой велели доказать безопасность, доказывает безопасность. Ещё пачка похороненных

дальше знания класса пополняются из ошибок прогонов: «переменная стоит в запросе» само по себе ещё не уязвимость; каждый разобранный промах превращается в новое правило

тезаурус: sink (сток) — вызов, где данные становятся командой (запрос к БД) · source — место входа данных пользователя · «грязная» переменная — из недоверенного источника и без защиты · экранирование — обработка спецсимволов, чтобы они не ломали запрос; действует только для значения в кавычках · guided decoding — сервер не выпускает ответ вне заданной схемы · Confirmed / фолз / Unknown — три вердикта: реальная / ложная / не знаю

TRACE НА ЖИВОМ ПРИЛОЖЕНИИ: 6 341 SQL-ИНЪЕКЦИЯ

17–18.09.2026 · заброшенное PHP-приложение, 45 913 сработок одного сканера, семейство инъекций 9 589; первое семейство trace-линии — SQL-инъекции (CWE-89)

ПРОБЛЕМА: УЛИК В КАРТОЧКЕ НЕТ

3,7 % карточек сработки показывают, откуда взялась переменная запроса (152 из 4 138); сам опасный вызов виден в 76 %

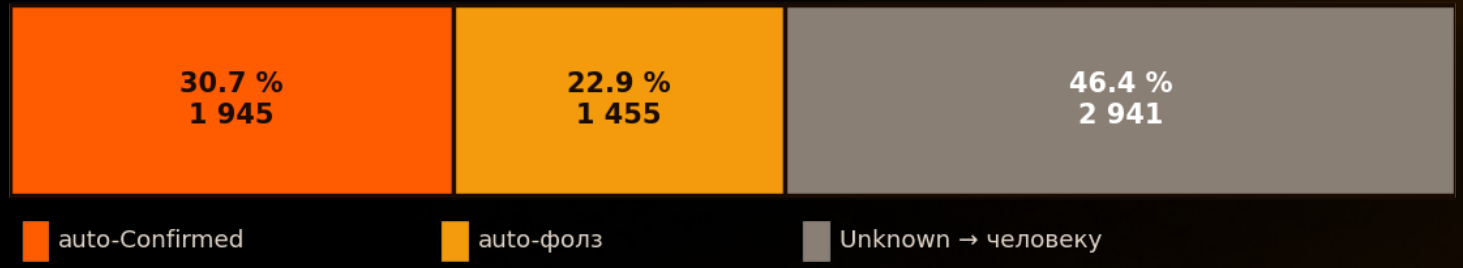
от вызова до присваивания переменной — медиана 2 138 строк, файлы до 16 528 строк

ЛЕЧЕНИЕ: ВОССТАНОВИТЬ ДЕРЕВО ИСХОДНИКОВ

188 файлов · 594 тыс. строк из выгрузок LLM-портала; покрыто 88 % SQL-сработок (5 560 из 6 341)

откуда берётся переменная: T1 в том же файле 50 % · T2 параметр функции 3 % · T3 из базы (второй порядок) 19 % · T4 файл вне дерева 28 % › Unknown

ДВИЖОК БЕЗ МОДЕЛИ ПО ВСЕМ 6 341 · резолвер (находит, откуда пришла каждая переменная) + gate (таблица правил G0–G12) · 237 с



НА 216 РАЗМЕЧЕННЫХ (29 эталонов + слепая переразметка 117 спорных + 70 случайных):

0
похоронено (реальная › фолз)

0,943
полнота по реальным: 82 из 87 › Confirmed

1,000
точность Confirmed (Уилсон 0,955)

60 / 60
канареек целы; в прогонах с моделью 69/69 и 40/40

Слепая переразметка 187 находок (я размечал код, не видя вердиктов движка): 12 из 15 «False Positive» агрегатора — реальные инъекции; три паттерна на всё приложение — помощник поиска `search_get()`, ключ массива `$_POST['in_edit']` прямо в запрос, проверка `preg_match` без якорей ^ и $. Найдено 9 дефектов резолвера — все закрыты парными тестами

Три прогона с моделью 14B: только карточка сработки — модель закрыла как фолз 17 из 36 реальных уязвимостей; с деревом — похоронено 0, и модель не добавила ни одного вердикта к движку (фолз 215/215, Confirmed 308/308 совпали); с кодом вызывающих функций — Unknown на параметрах функций 94 % › 52 %

тезаурус: сток (sink) — опасный вызов, где переменная становится частью SQL-запроса · резолвер — программа, находящая, откуда пришла каждая переменная · gate — таблица правил, выносящая вердикт · второй порядок — значение пришло из базы, куда его мог положить атакующий · Уилсон — нижняя граница 95 % доверительного интервала · Confirmed — вердикт «реальная уязвимость» · рука — вариант прогона с другой подачей контекста

ДВЕ ЛИНИИ РЯДОМ: ЧТО ОБЩЕЕ, ЧТО РАЗНОЕ

один и тот же режим «улики › gate», но улика, глубина и права автомата — разные

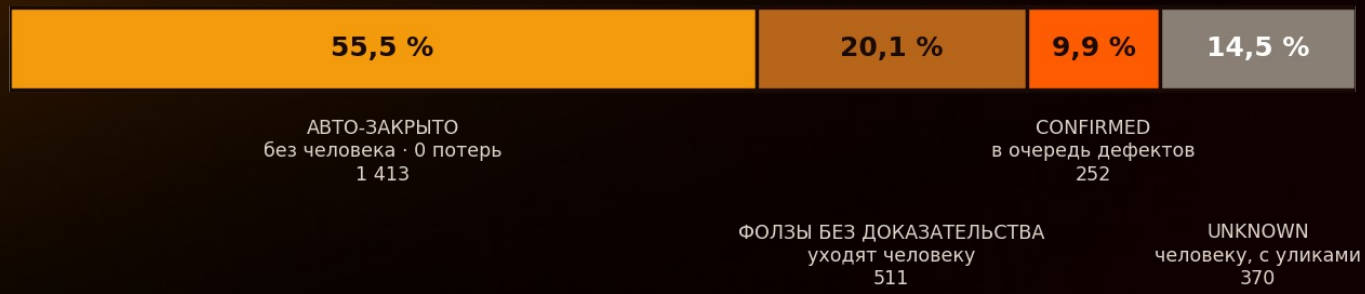
	PROPERTY-ЛИНИЯ	TRACE-ЛИНИЯ
единица доказательства	факт в строке + его контекст (роль файла, среда, роль значения)	путь данных source › sink со всеми преобразованиями
глубина улик	D0–D5 по подтипу: от байтов до окружения; чаще всего хватает карточки сработки	всё дерево исходников: четыре варианта происхождения переменной T1–T4 (слайд 16); карточки не хватает (3,7 %)
что делает модель	читает код и показывает пальцем: роль значения, назначение, безопасное условие — с цитатой	перечисляет каждую переменную запроса: откуда, какой строкой, защита, позиция; на SQL живого приложения не добавила к движку ни одного вердикта
что значит «не нашёл»	NOT FOUND ≠ DISPROVED › Unknown; P4/P6 — много Unknown по замыслу	источник вне дерева › Unknown; второй порядок (из БД) › человеку
что разрешено автомату	только закрыть как фолз; дефект объявляет человек (учётные данные в опасном месте — всегда человеку)	авто-Confirmed по правилу G6 и авто-фолз по G5 — на заброшенном приложении; в боевом — дефект через человека
типичная ошибка модели	закрыть «не вижу пути к стоку»; поверить пути test/ у живого токена	придумать санитайзер; засчитать экранирование не в той позиции запроса

ОБЩЕЕ: gate по контракту улик, цитаты проверяются посимвольно, канарейки в каждом прогоне, рубильник на класс, Unknown — полноценный результат, журнал прогона — источник всех цифр

тезаурус: санитайзер — функция-очиститель входных данных (экранирование, приведение к числу) · контракт улик — заранее описанный список того, что должно быть предъявлено для вердикта · G5 / G6 — правила gate: все переменные безопасны / все грязные доказаны · D0–D5 — глубина улик (слайд 10) · T1–T4 — четыре варианта того, откуда пришла переменная запроса: в том же файле / параметр функции / из базы / файл вне дерева (слайд 16) · P4 / P6 — подтипы «обязанность» и «окружение» · рубильник — одна заданная канарейка выключает класс

ЭКЗАМЕН: 2 546 РЕШЕНИЙ ЛЮДЕЙ

2 546 сработок обоих типов, уже разобранных людьми; система разобрала их вслепую — человеческих ответов не видела. Из 2 902 исходных исключены 88 закрытых служебной автоматикой и 268 решений соавтора системы



85,5 % сработок получили определённый ответ — «фолз» или «реальная» (покрытие)

87,8 % ответов совпали с решением людей (из 2 176 отвеченных)

96,5 % вердиктов «фолз» верны (1 856 из 1 924) — главный показатель: фолзы уходят в автозакрытие

0 из 370 реальных уязвимостей среди Unknown — всё сомнительное ушло человеку

РАЗОШЛИСЬ — 266

68 похороненных — реальные, закрытые как фолз: расплата за доверие всем вердиктам модели подряд; 46 из них — один боевой токен доступа в тестовых файлах (следующий слайд)

198 перестраховок — фолзы, отданные человеку как реальные: стоят минут ревьюера, дыр не создают

Консервативный режим: автомату разрешены только классы, где на замере не потеряно ничего, плюс блок-лист на проблемные (CWE-257, CWE-200, CWE-325) › 55,5 % очереди без человека при нуле наблюдаемых потерь. И это уже не стенд: 97 202 сработки production-масштаба прошли через модели в июле

тезаурус: покрытие — доля сработок с определённым ответом · Confirmed — вердикт «реальная», уходит в очередь дефектов · Unknown — «не знаю», уходит человеку с уликами · служебная автоматика — закрытия скриптами, без человека · блок-лист — классы, где автозакрытие выключено · production-масштаб — живой поток, не стенд

ГДЕ МЫ СПОТКНУЛИСЬ

три грабли за полгода — и все три поймали перепроверки; интуиция промолчала

1

НАРЕЗКА ДАННЫХ

на смеси проектов 93,8 % верных › внутри каждого проекта 41,7–60 %, на Go/Ruby — 0 %: скил подогнан под PHP. Вторая модель — та же яма: похороненные совпадают 6/6 и 5/6 — ошибка системная, смена модели не лечит. Лечение: мерить внутри каждого проекта

2

БОЕВОЙ ТОКЕН В ТЕСТОВЫХ ФАЙЛАХ

один настоящий токен доступа × 46 сработок; модель поверила пути: «placeholder в тестовом конфиге», уверенность 1.0 — все 46 похоронены. Это P6: истина вне кода. Лечение: алгоритмическая проверка «жив ли токен» на ступени 0; пометка test/ — подсказка, не доказательство

3

СБОРЩИК МОЛЧА РЕЗАЛ СКИЛ

лимит 16 800 символов, скил дорос до 22 800 — середину вырезал без предупреждения; новые скилы «ухудшили» качество втрое. Ответ нашёлся в журнале доставки. Подняли лимит › класс CWE-366 (гонка потоков): 0,010 › 0,810. Лечение: порог на доставку скила

Снаружи цифра, которая рассыпается при смене нарезки, и скил, который не доехал, выглядели правдоподобно. Предохранители в первую очередь ловят нас самих

тезаурус: нарезка — способ разбить выборку для замера (по проектам, по языкам) · скил — промпт (текст задания), заточенный под класс дефекта · placeholder — заглушка вместо настоящего значения · токен доступа — ключ, по которому система пускает без пароля · сборщик — программа, склеивающая промпт из частей · P6 — подтип «окружение»: истина вне кода

«Неважно, как часто ты падаешь. Важно, как часто ты поднимаешься» — Джейсон Стетхем

СЕГОДНЯ МЫ УЗНАЛИ МНОГО НОВОГО

шесть вещей — и все про структуру; про размер модели ни одной

1 Сработки бывают двух сортов. Property доказываются проверкой факта, trace — проверкой пути данных. Один вопрос «покажи путь» на всех — это потеря 9 из 10 реальных property-уязвимостей на ровном месте

2 Сначала форма доказательства — потом вопрос. Подтип P0–P6 и глубина улик D0–D5 объясняют ошибки триажа лучше, чем CWE или миллиарды параметров; CWE говорит «о чём», сигнатура — «как доказывать»

3 Модель находит — программа проверяет. Цитата, не найденная в коде посимвольно, не существует; вердикт выносит gate по письменной политике. Семейства: 247 канареек, задавлено 0; эталоны — 0 похороненных в exc, secrets, sensitive и SQL;

4 NOT FOUND ≠ DISPROVED. Unknown — механизм безопасности: 0 из 370 реальных спрятались в Unknown; Unknown — норма

5 Уверенность модели — не пропуск в автозакрытие: ответы с confidence 1.0 верны реже, чем с 0.8. Пропуск — только доказательство, проверенное алгоритмически

6 Что сколько даёт. Пречеки (правила до модели) и постчеки (проверки после): ¼–⅔ закрытий с точностью 0,99 без модели. Модель одна: 38 % закрытий и 4 задавленные канарейки из 29; на trace по одной карточке — 17 из 36 реальных похоронены. Связка правила + модель + gate: 67 % и 0 канареек — ни одна половина не заменяет другую

НЕ ЗАСТАВЛЯЙТЕ МОДЕЛЬ УГАДЫВАТЬ. СНАЧАЛА РЕШИТЕ, КАК ВЫГЛЯДИТ ДОКАЗАТЕЛЬСТВО, — ПОТОМ ЗАСТАВЬТЕ СИСТЕМУ ПРОВЕРЯТЬ.

тезаурус: P0–P6 / D0–D5 — подтипы property-сработок и глубина улик (слайд 10) · сигнатура доказательства — набор признаков правила сканера: как его доказывать · gate — ограничитель, выносящий решение по письменной политике · канарейка — подмешанная сработка с реальным дефектом · confidence — уверенность, которую модель заявляет сама · пречеки / постчеки — алгоритмические проверки до вызова модели / после её ответа · агрегатор — система, где копятся сработки всех сканеров

ЧТО ДАЛЬШЕ

ближайшие шаги обеих линий — осень 2026

1. Trace: перепись — 13 161 SQL-инъекция, ~4 600 в PHP-приложениях семи других команд — нужны их деревья исходников; остальные стоки того же приложения (вывод в HTML, файлы, команды ОС); динамический оракул — подмена функции запроса к базе и сравнение деревьев запроса с атакующим значением и без
2. Роутер по сигнатуре доказательства для всех 180 property-классов: таблица правило › подтип › глубина › обязательные улики › безопасные условия › роль LLM — вместо 180 новых промптов
3. Оркестратор: доказанные лёгкие property-классы — малым моделям 1,5–3B (13 классов = 99 % первой очереди), тяжёлые trace — большим, под мощность железа
4. Структурная генерация скилов по контракту, бюджет места, схема ответа под класс; защита по обоснованию — «критичный риск» в обосновании понижает фолз до Unknown

тезаурус: перепись — разбор всех сработок агрегатора по типу и приложению · дерево исходников — полные файлы приложения · сток — опасный вызов, где данные становятся командой · динамический оракул — проверка исполнением кода: подставить атакующее значение и посмотреть, изменилась ли структура запроса · роутер / сигнатура — слайды 9–10 · оркестратор — распределитель сработок между моделями · скил — промпт под класс дефекта · схема ответа — список и порядок полей, которые модель обязана заполнить

«Neo, sooner or later you're going to realize, just as I did, there's a difference between knowing the path and walking the path» — Морфеус

СПАСИБО ЗА ВНИМАНИЕ!

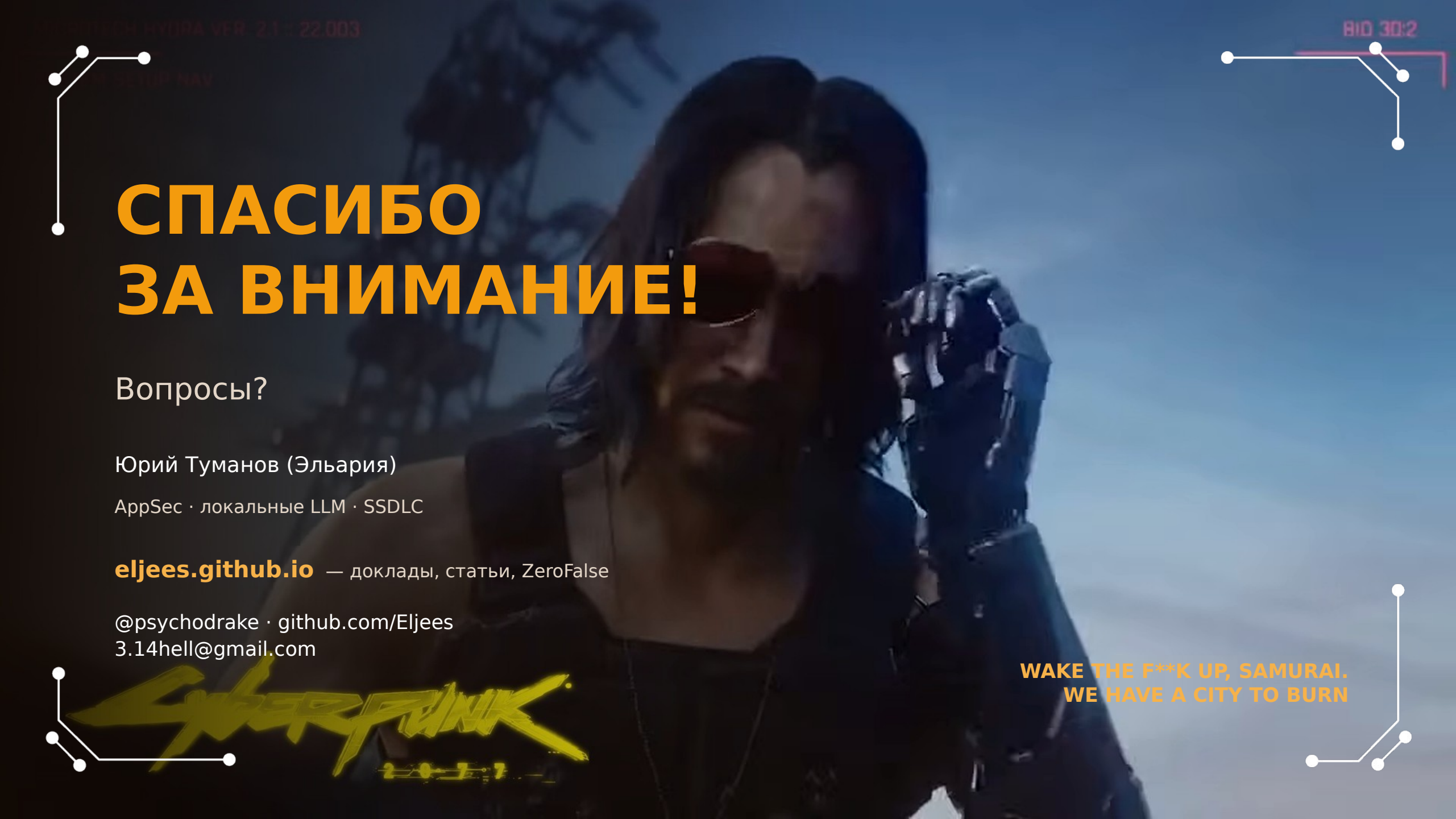
Вопросы?

Юрий Туманов (Эльария)

AppSec · локальные LLM · SSDLC

eljees.github.io — доклады, статьи, ZeroFalse

@psychodrake · github.com/Eljees
3.14hell@gmail.com



А3 · ПАРК МОДЕЛЕЙ: 1,5–24 МЛРД ПАРАМЕТРОВ

весь парк — локальный, в 16 ГБ видеопамяти одной RTX 4080 (два сервера запуска моделей); B — миллиарды параметров; q4 / AWQ — квантование

Qwen2.5-Coder-1.5B (q4) · единственная живёт на смартфоне: живой триаж на телефоне — вердикты совпали с видеокартой 37/39 · медиана 26,6 с против 3,5 с

Qwen2.5-Coder-3B (q4) · эксперименты и дешёвые классы; после автонастройки промптов точность на классе CWE-732 выросла с 0,667 до 0,857; хоронит меньше канареек, чем 1.5B

Qwen2.5-Coder-7B (q4) · массовые прогоны; самооценке не верим: за неделю 130 ответов с уверенностью 0.0 и 154 с 1.0; подтверждать trace-уязвимости ей запрещено

Qwen2.5-Coder-14B (AWQ) · основная модель главного замера и всех пяти семейств property-линии · 4,5–9 с на решение

Foundation-Sec-8B · дообучена под безопасность; просишь JSON — а она тебе поэму. Битый JSON 26 %; подняли лимит длины ответа › 4,8 %

Devstral-Small-2 (24B) · самая точная из локальных: на SQL-инъекциях 93,8 % верных вердиктов против 76,2 % у 14B (но см. граблю № 1)

Qwen2.5-72B-Instruct · антипример: доступ через LLM-портал — формат ответа не зафиксировать › уверенные фантазии; на trace-переразметке расходится с движком в 30–45 %

обе сильные модели хоронят ОДИН И ТОТ ЖЕ набор находок (совпадение 6/6 и 5/6) — смена модели не лечит системную ошибку; нарезка данных и процедура важнее размера

тезаурус: сервер запуска моделей — программа, которая крутит модель на видеокарте и отвечает по сети · q4 / AWQ — сжатие модели до 4 бит (квантование), чтобы влезла в память · JSON — структурированный формат ответа · trace-уязвимость — сработка типа «путь данных» · дообучение — доводка готовой модели на своих данных

A1 · ФОРМУЛЫ И ПРАВИЛА UNKNOWN

Матрица исходов: a = real › Conf · b = real › FP (ПОХОРОНЕНА) · c = real › Unk · d = fp › Conf · e = fp › FP · f = fp › Unk

`P = a / (a + d)` — точность вердиктов «реальная» (precision Confirmed)

`Rб = (a + c) / Np` — доля НЕ потерянных реальных (safety-recall; Unknown = сохранено) · `Rд = a / Np` (Unknown = промах)

`Ротс = e / (e + b)` — точность авто-отсева (доля правды среди закрытых как фолз) · `Ротс = e / Nл · Q = e / N`

`coverage = решённые / N · abstention = (c + f) / N · риск = 100·b / (e + b)` — потерь на 100 авто-отсевов

`AutocloseRisk`$_{100}$`, SafetyRecall, Precision_AutoFP` — те же величины в терминах исследования подтипов

Unknown: НИКОГДА не «правильный ответ»; при сверке с людьми — несовпадение; в боевую систему не пишется. Gate: Confirmed — целостность в порядке и все обязательные «плохие» улики PROVED; FP — целостность в порядке и безопасное условие PROVED; Unknown — иначе, в том числе при любом CONFLICT

Уверенность по подтверждениям: 0 подтверждённых цитат › ≤ 0,55 · 1 › ≤ 0,75 · закрывать можно от 0,85 — одной цитаты для закрытия не хватает никогда

Доверительные интервалы: нижняя граница Уилсона 95 % — при малых n широкая; публикуется только с N

parse / timeout / llm-error — отдельный класс исходов, в метрики качества не входит; на 2 902 прогонах главного замера — 0,0 %

A2 · ПОЛНЫЕ МАТРИЦЫ ГЛАВНОГО ЗАМЕРА

ВСЕ РЕШЕНИЯ ЛЮДЕЙ (n = 2 814):

	люди: FP	люди: Conf
система FP	1 969	166
система Conf	221	84
система Unk	374	0

совпадение 73,0 % · без Unknown 84,1 % · точность «фолз» 92,2 % · Unknown 13,3 %

ПОСЛЕ ОЧИСТКИ — только ревьюеры, не видевшие ответов системы (n = 2 546):

система FP	1 856	68
система Conf	198	54
система Unk	370	0

совпадение 75,0 % · без Unknown 87,8 % · точность «фолз» 96,5 % · Unknown 14,5 %

срез соавтора системы (n = 268): совпадение 53,4 %; 98 из 166 «Conf › FP» полного набора — его решения · 88 сработок закрыты служебной автоматикой — исключены

A4 · ИЗЯЩНЫЕ МОДЕЛЬКИ И PROPERTY-ПЕРЕПИСЬ

Перепись: метаданные всех 1 004 440 сработок агрегатора, классификация по типу локально › 934 572 property (93 %)

Корпус: 10 825 property-сработок со всех 180 классов, у каждой — вердикт человека · скил на каждый класс (87 боевых + 93 синтезированных) · 2 модели × 2 сборки промпта = 43 300 вердиктов

Итог: первая очередь — 17 классов = 381 366 сработок = 40,8 % property-потока (порог: ноль похороненных у всех четырёх конфигураций, точность закрытия ≥ 0,90)

Экзамен малых моделей (тот же срез побайтово: 1 324 фолза + 101 канарейка · Qwen2.5-Coder 1.5B / 3B / 7B):

1 · гипотеза «возьмём модель побольше» опровергнута: 7B вдвое больше 3B — и хуже во всех трёх схемах ответа: через порог 0 классов из 17 (у 3B — 8)

2 · решает схема ответа: «сначала дословные цитаты из кода, потом вердикт» › доля закрытых фолзов 0,801 против 0,345 у мягкой схемы

3 · лучшие связки «класс › модель + схема»: 13 классов = 99,4 % объёма первой очереди на моделях 1,5–3B · 0,7–1,7 с на сработку против ~7 с у 14B

4 · проверка на прочность: малые модели похоронили до 48 канареек из 101 — отличать настоящий дефект от ложного сами не умеют; в бою первая же похороненная канарейка останавливает класс

5 · защита по обоснованию: «критичный риск» в обосновании + вердикт «фолз» › понижение до Unknown; на непросмотренном срезе — треть похороненных возвращена за 6 п.п. полноты

тезаурус: перепись — разбор метаданных всех сработок агрегатора по типу · скил — промпт под класс дефекта · сборка промпта — вариант его компоновки · первая очередь — классы, допущенные к автозакрытию первыми · канарейка — подмешанная сработка с заведомо реальным дефектом · схема ответа — список и порядок полей, которые модель обязана заполнить · защита по обоснованию — тревожное обоснование модели понижает её же фолз до Unknown

A5 · СРЕЗ ПО ПРОЕКТАМ: SQL-ИНЪЕКЦИИ (CWE-89)

Devstral-24B, процедура v3, полные корпуса приложений (порог приёмки P ≥ 85 %, Rб ≥ 95 % — задан ДО прогонов). baseline = доля фолзов в корпусе

```
Проект A (PHP)      n=40  P 52.9  Rб 100.0  похоронено 0  Ротс 100.0  baseline 50.0
Проект B (PHP)      n=40  P 57.1  Rб 100.0  похоронено 0  Ротс 100.0  baseline 52.5
Проект C (PHP)      n=40  P 60.0  Rб  75.0  похоронено 5  Ротс  64.3  baseline 55.0
Проект D (Go/Ruby)  n=30  P 41.7  Rб  60.0  похоронено 6  Ротс   0.0  baseline 76.7
```

Qwen-14B на тех же корпусах: P 51,9–61,5 · похороненные пересекаются с Devstral почти полностью — ошибка системная

Живое приложение trace-линии (17–18.09, движок v4): формы улик по 6 341 SQL-сработке — T1 (источник в файле) 3 172 · T2 (параметр функции) 190 · T3 (второй порядок) 1 222 · T4 (вне дерева) 1 757

Проба по дереву (без модели): HIGH 41 % (идентификатор / без кавычек 1 842 + сырой запрос 440) · MEDIUM 22 % · LOW 22 % · NO_SINK 15 %; проба и аналитик не расходятся 12/12, 72B расходится 30–45 %

Gate по правилам: G6 Confirmed 1 945 · G5 FP 1 234 · G2 «не сток» 218 · G8 второй порядок 1 432 · G0 вне дерева 1 009 · G11 переменная не разрешена 379 · G9 сессия 77 · G12 44 · позиция инъекций: в кавычках 81 %, идентификатор 12 %, без кавычек 7 %

Три руки с моделью Qwen-14B: B0 только карточка — модель закрыла как FP 17 из 36 реальных (47 %); B1 дерево + окна — DS 0, TP-recall 35/36, движок закрывает 215/215 FP и подтверждает 308/308 Confirmed руки B1 (модель вердиктов не добавляет); B2 + тела помощников и вызывающие — Unknown на T2 94 % › 51,7 % при DS 0; отвергнутых цитат модели 15–17 % (норма ≤ 5 % не пройдена — вердикт держится на фактах движка)

Разметка агрегатора как эталон слаба в обе стороны: 12 из 15 «False Positive» при Confirmed движка — реальные инъекции; 3 — копии файлов с датой/_old, достижимость неизвестна (G4 › to_verify)

тезаурус: P — точность вердиктов «реальная», Rб — доля не потерянных реальных, Ротс — точность авто-отсева (формулы в A1) · baseline — доля фолзов в корпусе: столько закрыл бы вердикт «всё фолз» · корпус — все сработки класса в одном приложении · рука (B0–B4) — вариант прогона с другой подачей контекста · DS — опасное закрытие реальной уязвимости · T1–T4 — четыре варианта происхождения переменной (слайд 16)

A6 · УВЕРЕННОСТЬ МОДЕЛИ — НЕ ПРОПУСК

точность ответов по заявленной самой моделью уверенности (главный замер, после очистки):

confidence 0.8 › 92 % верных (n = 1 687)

confidence 1.0 › 76 % верных (n = 310) — максимально уверенные ответы ХУЖЕ

порог «пускать в автозакрытие только confidence ≥ 0.9» дал бы 13,5 потерь на 100 автозакрытий — вчетверо хуже простого приёма всех фолз-вердиктов

причина кластера самоуверенных ошибок: 46 копий одного и того же боевого токена, закрытых как «placeholder» с уверенностью 1.0

поэтому в property-линии уверенность пересчитывается по числу независимых подтверждённых цитат, а поле «confidence» модели — только диагностика

A7 · ВОСПРОИЗВОДИМОСТЬ: ГДЕ ЛЕЖАТ ЦИФРЫ

Каждая цифра доклада — из журнала прогона: 07_claims_ledger.csv (источник, формула, N, допущения) для главного замера; протоколы Э3/Э4 и отчёты семейств — для property-линии; E3_PROTOCOL_INJ — для trace

Property-линия (по семействам): план методики (Э0–Э5, решения P1–P9) › срез с вердиктами людей › Э3 офлайн с канарейками › Э4 сухой прогон на живой очереди › отчёт; повторяемость меряется вторым полным прогоном (http: 1 107 из 1 107 решений совпали)

Гейты приёмки (пример http, 11 проверок): доставка одним хешем · 0 опасных закрытий на эталоне · канарейки 100 % · точность закрытий ≥ 0,95 · снятие рутины ≥ 80 % · Unknown ≤ 25 % · ошибки формата ≤ 0,5 % · отвергнутых цитат ≤ 5 % — 4 из 11 не пройдены — и это записано в отчёте

Trace-линия: срез 15 903 записей заморожен, дерево исходников с хешем, движок 15 парных тестов; руки эксперимента: B0 карточка · B1 дерево + окна · B2 + помощники и вызовы · B3 без модели · B4 72B; наборы D1 (29 + 117 слепых) · D2 агрегатор · D3 72B · D4 60 канареек

Исследование подтипов: deep-research-отчёт 06.09.2026 — таксономия P0–P6, шкала D0–D5, сигнатура из 11 признаков, контракт улик с четырьмя состояниями PROVED / DISPROVED / UNKNOWN / CONFLICT; карта CWE › подтип — маппинг по умолчанию, проверяется на корпусе

Что просить у меня после доклада: планы семейств, протоколы, сводные таблицы по эталонам (каждая сработка: вердикт человека, решение gate, ответ модели, прежний конвейер) — без исходников заказчиков

тезаурус: журнал прогона — файл, куда конвейер пишет каждое решение и его основания · леджер — таблица «цифра — источник — формула — N» · протокол — документ об этапе эксперимента с таблицами и приёмочными проверками · гейты приёмки — заранее записанные пороги, которые прогон обязан пройти · хеш — контрольная сумма дерева исходников: одинаковый хеш = одинаковый вход · Э0–Э5 / P1–P9 — этапы плана / решения, которые принимает автор