https://doi.org/10.51885/3107-2755_JISC_2026_2_2
МРНТИ 81.93.29

МЕЖУРОВНЕВАЯ КОРРЕЛЯЦИЯ ЖУРНАЛОВ LINUX И eBPF-ТЕЛЕМЕТРИИ ПРИ ЛОКАЛИЗАЦИИ ПЕРВОПРИЧИН ОТКАЗОВ

АҚАУЛАРДЫҢ ТҮПКІ СЕБЕПТЕРІН ЛОКАЛИЗАЦИЯЛАУ КЕЗІНДЕ LINUX ЖУРНАЛДАРЫ МЕН eBPF ТЕЛЕМЕТРИЯСЫН ДЕҢГЕЙАРАЛЫҚ КОРРЕЛЯЦИЯЛАУ

CROSS-LAYER CORRELATION OF LINUX LOGS AND eBPF TELEMETRY FOR ROOT-CAUSE LOCALIZATION

А.В. Арефьев [1], **В.Г. Фарафонов** [1], **А.Р. Тулегенов** [2*]
[1] Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП), кафедра прикладной математики, Санкт-Петербург, Россия
[2] ТОО «МАЭК», Астана, Казахстан
*Автор-корреспондент: Тулегенов А.Р., e-mail: kaz-arman-tulegenov@yandex.ru.

Ключевые слова:
межуровневая корреляция, eBPF, журналы событий, анализ первопричин, телеметрия Linux, граф событий, причинные кандидаты, наблюдаемост реагирование на инциденты информационной безопасности.

АННОТАЦИЯ
Исследовать межуровневую корреляцию журналов и телеметрии расширенного фильтра пакетов eBPF для локализации первопричин отказов Linux-систем. Методы. Канонические события связывались по устойчивой идентичности процессов и сокетов; причинные кандидаты ранжировались после проверки совместимости свидетельств с механизмом отказа. Проведены четыре эксперимента: пассивное наблюдение, углублённая трассировка, внесение трёх известных отказов и проверка четырёх ранее не использовавшихся механизмов. Результаты. Рост объёма телеметрии не гарантировал улучшения диагностики. На известных механизмах фильтрация повысила долю правильных первых ответов с 0,667 до 1,000. На 20 окнах новых отказов модели E-R и F0 дали одинаковую точность локализации домена и покрытие правильной причины: 9/20=0,450. Число кандидатов сократилось с 4035 до 19. Заключение. Наблюдаемое преимущество состоит в сокращении пространства поиска при неизменном покрытии в данной выборке. Результат не доказывает безопасность фильтра вне эксперимента. Метод применим как слой корреляции диагностических свидетельств для разбора инцидентов, но не как самостоятельный детектор атак.

Түйінді сөздер:
деңгейаралық корреляция, eBPF, оқиғалар журналдары, түпкі себептерді талдау, Linux телеметриясы, оқиғалар графы, себептік кандидаттар, бақыланғыштық, ақпараттық қауіпсіздік инциденттеріне ден қою.

ТҮЙІНДЕМЕ
Linux жүйелеріндегі ақаулардың түпкі себептерін анықтау үшін қолданбалы журналдар мен кеңейтілген пакеттік сүзгі eBPF телеметриясын деңгейаралық корреляциялау әдісін зерттеу. Әдістері. Бақылаулар канондық оқиғаларға қалыптандырылды; құрылымдық байланыстар процестер мен сокеттердің тұрақты сәйкестендіргіштері бойынша құрылды. Себептік кандидаттар диагностикалық деректер ақау механизмімен үйлескеннен кейін ранжирленді. Төрт эксперимент жүргізілді: пассивті бақылау, терең трассалау, үш белгілі ақауды енгізу және бұрын қолданылмаған төрт механизмді тексеру. Нәтижелері.

Телеметрия көлемінің өсуі диагностика сапасын міндетті түрде жақсартпады. Белгілі механизмдерде сүзгілеу бірінші жауаптың дұрыстығын 0,667-ден 1,000-ге арттырды. Жаңа ақаулардың 20 терезесінде E-R және F0 модельдері үшін доменді анықтау дәлдігі мен дұрыс себепті қамту бірдей болды: 9/20=0,450. Кандидаттар саны 4035-тен 19-ға азайды. Қорытынды. Осы іріктемеде іздеу кеңістігі қамту өзгермей қысқарды; нәтиже сүзгінің эксперименттен тыс қауіпсіздігін дәлелдемейді. Әдіс инциденттерді талдауға арналған диагностикалық деректерді байланыстыру қабаты ретінде қолданылуы мүмкін, бірақ шабуылдарды өздігінен анықтамайды.

keywords:
cross-layer correlation, eBPF, event logs, root-cause analysis, Linux telemetry, event graph, causal candidates, observability, security incident response.

ABSTRACT

To investigate cross-layer correlation of application logs and extended Berkeley Packet Filter (eBPF) telemetry for system-event graph reconstruction and root-cause localization in Linux systems. Methods. Observations are normalized into canonical events; structural relations rely on stable process and socket identities, while causal candidates are admitted to ranking after compatibility of the available evidence with the observed mechanism is checked. The method was evaluated in four consecutive experiments: passive observation of two long-running hosts, deep tracing, controlled injection of three known faults, and held-out validation on four previously unseen mechanisms. Results. Increasing telemetry volume did not monotonically improve localization. In the original experiment, the richest model and the log-only model both achieved mean reciprocal rank (MRR)=0.667, with mean candidate-set sizes of 124.83 and 1.62. For known mechanisms, mechanism-aware admissibility increased Top-1 from 0.667 to 1.000. On held-out mechanisms, domain Top-1 and candidate coverage were 0.450 (9/20) for both E-R and F0. Across the same 20 fault windows, E-R generated 4,035 candidates (mean=201.75), whereas F0 generated 19 (mean=0.95). Thus, the reproducible effect of F0 on this sample is a substantial reduction of the candidate search space at unchanged coverage=0.450; the result does not imply high diagnostic completeness or establish filter safety outside the evaluated data. Conclusion. The principal observed limitations were related to evidence availability, causal representation, and candidate generation. For information-security use, the method should be interpreted as an evidence-correlation layer for incident triage and response rather than as a standalone attack detector.

ВВЕДЕНИЕ

Linux-системы формируют журналы, метрики ресурсов и телеметрию процессов, системных вызовов и соединений. Прикладной журнал отражает представление приложения, а kernel telemetry — действия процесса, не всегда сохраняя смысл операции. Поэтому локализация первопричины требует достоверных межуровневых связей; увеличение числа источников само по себе не обеспечивает диагностику.

Распределённая трассировка связывает события контекстом запроса: X-Trace восстанавливает его путь (Fonseca et al., 2007), а Mint уменьшает затраты хранения и передачи за счёт общих частей трасс (Huang et al., 2025). Однако legacy-приложения и закрытые комплексы часто не имеют request/trace identifier и не допускают изменения кода. Для них зависимости приходится восстанавливать по системным признакам.

eBPF позволяет с относительно малыми накладными расходами наблюдать процессы, системные вызовы и сетевые состояния. Без устойчивой идентичности процесса, сокета или сервиса дополнительные события могут увеличивать число совпадающих по времени гипотез. Здесь исследуются информативность межуровневых признаков и границы их применимости для RCA.

Метод формирует evidence layer для triage, incident response и forensic-анализа критичных сервисов. Завершение процесса, недоступность сервиса, TCP reset/refusal, сбои разрешения имён и истощение ресурсов могут сопровождать как эксплуатационные сбои, так и инциденты ИБ. Для различения причин нужны audit/authentication logs, средства сетевого обнаружения и контекст изменений; метод не классифицирует атаки и злонамеренный умысел.

Цель — экспериментально исследовать корреляцию прикладных журналов и eBPF-телеметрии для реконструкции графа событий и локализации первопричин. Проверяются возможность связывания приложения, процесса, syscall, сокета и соединения без обязательной распределённой трассировки и модификации приложения, влияние дополнительных источников и уменьшение числа ложных кандидатов посредством механизм-ориентированной проверки evidence.

В первых двух экспериментах изучались признаки корреляции и стоимость углублённой трассировки. В третьем вносились три класса отказов; заранее зафиксированный ground truth был недоступен коррелятору. Затем выполнялись offline-абляция прикладных полей и сравнение A–F. В четвёртом замороженный фильтр проверялся на четырёх новых механизмах, чтобы отделить эффект подгонки от воспроизводимого свойства.

ЛИТЕРАТУРНЫЙ ОБЗОР

Методы наблюдаемости можно условно разделить на несколько групп. Первая связана с распределенной трассировкой и передачей контекста. X-Trace формализовал межкомпонентное связывание событий по контексту запроса (Fonseca et al., 2007), а современные работы по экономичному трассированию показывают возможность уменьшать стоимость сбора, не отказываясь от охвата запросов (Huang et al., 2025). Сильной стороной таких методов является явное причинное основание; ограничением — необходимость инструментирования приложения, библиотек или инфраструктуры передачи контекста.

Вторая группа основана на анализе журналов. Drain используется для потокового выделения шаблонов сообщений (He et al., 2017), а DeepLog рассматривает последовательности журналов как источник для обнаружения аномалий и диагностики (Du et al., 2017). Существенным ограничением остается то, что журнал содержит только те сущности, которые разработчик приложения решил записать. В исследуемых эксплуатационных журналах отсутствовали request identifier и сетевой endpoint, а связь сообщения с конкретным процессом часто восстанавливалась только по метаданным journald.

Третья группа ориентирована на автоматический RCA в микросервисных и облачных системах. MicroRCA, MicroHECL, Groot и CloudRCA используют графы зависимостей, метрики, журналы и топологию для локализации причин (Wu et al., 2020; Liu et al., 2021; Wang et al., 2021; Zhang et al., 2021). Более поздние мультимодальные методы объединяют метрики, трассы и журналы, применяя обучаемые представления и причинные структуры (Zhang et al., 2023; Lee et al., 2023; Yu et al., 2023; Zheng et al., 2024; Zhu et al., 2024; Zhao et al., 2026). Отдельно исследуются RCA при ограниченной наблюдаемости и динамических serverless-системах (Xie et al., 2024; Huang et al., 2024). Для таких методов качество результата определяется не только алгоритмом ранжирования, но и тем, какие сущности и отношения доступны до стадии ranking.

Отдельное направление связано с низкоуровневой наблюдаемостью. FUSE применяет eBPF для диагностики и подавления отказов в микросервисах (Ramachandran et al., 2023), а eBPF-enhanced observability позволяет получать системные признаки без обязательной языковой инструментализации (Sharma & Nadig, 2024). Наиболее близок к рассматриваемой задаче XPLOG: он объединяет eBPF-события, прикладные и системные журналы и сохраняет причинный порядок в распределенной среде (Satapathy et al., 2026).

Таким образом, использование eBPF, объединение журналов с системной телеметрией и построение графов сами по себе не рассматриваются в данной работе как новое решение. Современный обзор методов обнаружения аномалий и RCA подтверждает смещение исследований к мультимодальным данным, графовым моделям и объяснимой локализации причин (Barata et al., 2026). Настоящее исследование сосредоточено на измеряемом значении отдельных correlation signals, явном учете покрытия правильной причины множеством кандидатов и разделении ошибок, возникающих до ранжирования. Дополнительно проверяется, сохраняет ли механизм-ориентированная фильтрация свое действие после фиксации правил на ранее не использовавшихся механизмах отказа.

МАТЕРИАЛЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

Модель межуровневой корреляции. Пусть каждый наблюдаемый факт после нормализации представлен каноническим событием.

$$e_i = (t_i, h_i, s_i, \tau_i, p_i, q_i, a_i), \quad (1)$$

Структура межуровневого графа и типы связей, используемые для формирования причинных кандидатов, представлены на рисунке 1.

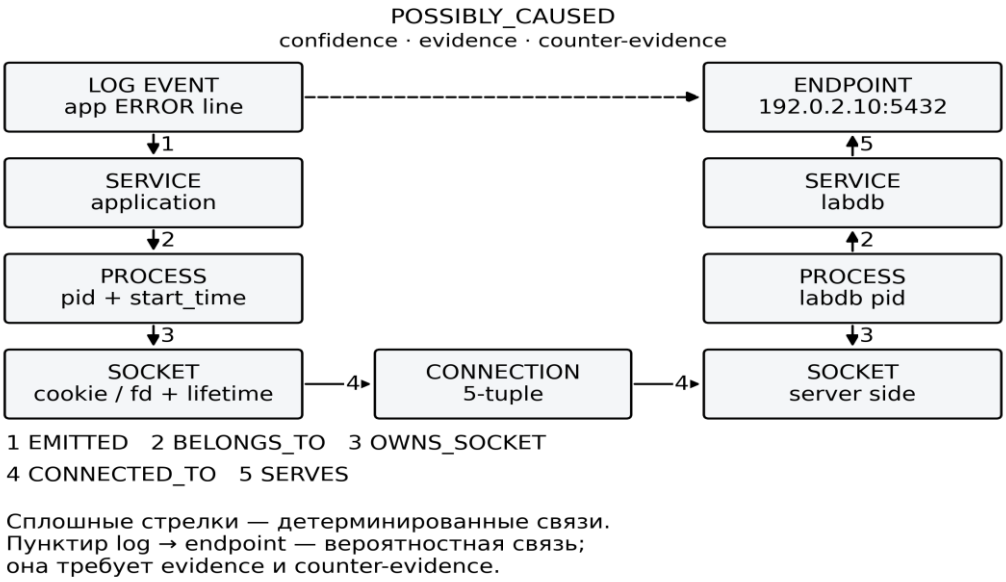


Рисунок 1. Пример межуровневого графа событий

Примечание - составлено авторами

где t_i — временная отметка, h_i — идентификатор узла, s_i — источник телеметрии, τ_i — тип события, p_i — идентичность процесса, q_i — идентичность сокета или сетевого отношения, a_i — набор типизированных атрибутов. В набор a_i включаются, в частности, системный вызов, длительность, errno, TCP state, уровень журнала, идентификатор

шаблона и прикладные поля. Исходная запись не уничтожается: нормализованное событие содержит ссылку на raw record.

PID не используется как самостоятельная идентичность процесса, поскольку числовой PID переиспользуется. В работе применяется составная идентичность

$$P = \langle H, PID, Tstart \rangle, \quad (2)$$

где H — узел, T_start — время запуска процесса. Аналогично числовой файловый дескриптор не считается идентификатором сокета. Для сокета используется либо kernel socket cookie, либо составная величина.

$$S = \langle P, FD, Topen \rangle. \quad (3)$$

Закрытие сокета завершает срок действия такого идентификатора. Это существенно при интенсивном переиспользовании FD, когда один и тот же номер соответствует нескольким соединениям в пределах короткого временного интервала.

Нормализованные события образуют ориентированный граф G=(V,E). Вершины соответствуют узлам HOST, SERVICE, PROCESS, LOG_EVENT, SYSCALL_EVENT, SOCKET, CONNECTION, ENDPOINT и RESOURCE_EVENT. Ребра делятся на структурные и вероятностные. К структурным относятся RUNS_ON, BELONGS_TO, EMITTED, OWNS_SOCKET, USES_FD, CONNECTED_TO и SERVES, если отношение подтверждено идентичностью и временем жизни объекта. Ребра DEPENDS_ON, CORRELATED_WITH и POSSIBLY_CAUSED считаются вероятностными и должны сопровождаться evidence, counter-evidence и величиной неоднозначности.

Для ограничения множества причинных гипотез вводится функция допустимости A(c,m), где c — кандидат причины, m — класс наблюдаемого механизма или симптома:

$$A(c, m) \in \{0, 1\}. \quad (4)$$

При A(c,m)=0 кандидат не передается на стадию ранжирования. В реализованной модели F проверяются два условия. Во-первых, тип кандидата должен быть способен породить наблюдаемый класс симптома. Во-вторых, для обвинения процесса или сервиса требуется структурное отношение владения: например, завершившийся процесс может рассматриваться как причина отказа endpoint только в том случае, если ранее наблюдалось, что этот процесс обслуживал данный endpoint. Тем самым временная близость рассматривается как контекст, но не как достаточное причинное основание.

Входом R-correlator является поток канонических событий. Evidence агрегируется в интервалы по 5,0 с. Интервал отмечается как аномальный, если значение превышает робастный baseline median+3·MAD, либо прикладная latency превышает тройную медиану, либо syscall latency превышает пятикратную медиану. Отмеченные интервалы, разделенные не более чем 15,0 с, объединяются в один incident. Merge window=15,0 с был выбран по распределению формы incidents, сохраненному из Experiment 3, а не по ground-truth labels.

Генератор использует шесть типов причинных кандидатов, образующих реализованный causal vocabulary: ENDPOINT_REFUSING, NETWORK_LATENCY_TO_ENDPOINT, APPLICATION_LATENCY_RISE, APPLICATION_ERROR_BURST, PROCESS_TERMINATED и ENDPOINT_CHURN. Оценки являются фиксированными формулами и не содержат обучаемых весов. Для ENDPOINT_REFUSING score=min(0,95; 0,35+0,12·log10(1+n)); для NETWORK_LATENCY_TO_ENDPOINT — 0,80; APPLICATION_LATENCY_RISE — 0,25; APPLICATION_ERROR_BURST — 0,30; PROCESS_TERMINATED — 0,85, если accept-records подтверждают обслуживание отказавшего endpoint данным процессом, и 0,20 в противном

случае; ENDPOINT_CHURN — min(0,60; 0,20+0,08·log10(1+n)), где n — число поддерживающих evidence events. В endpoint-failure evidence включаются только errno, классифицированные как реальные сетевые ошибки; EAGAIN и EINPROGRESS исключаются.

Перед ranking Model F классифицирует наблюдаемый симптом только по evidence, без ground truth, как SERVICE_FAILURE, CONNECTION_FAILURE, LATENCY или UNCLASSIFIED. Затем последовательно применяются три admissibility rules: type gate допускает только типы кандидатов, способные породить наблюдаемый класс симптома; ownership gate исключает PROCESS_TERMINATED без структурного подтверждения, что процесс обслуживал отказавший endpoint; endpoint-reality gate исключает endpoint-candidate, если соответствующий endpoint в данном окне фактически не отказал. При SERVICE_FAILURE сохранившийся PROCESS_TERMINATED получает единственную корректировку score +0,10 с ограничением 0,99. Других модификаций score нет.

После gate кандидаты ранжируются по убыванию score, затем по убыванию числа поддерживающих evidence events, затем лексикографически по subject; последнее правило обеспечивает детерминированный полный порядок. Пустое множество кандидатов допустимо и фиксируется как candidate-generation failure. Если правильная причина отсутствует среди кандидатов, rank считается неопределенным, а запуск дает нулевой вклад в Top-1, Top-3, MRR и candidate coverage, но не исключается из знаменателя.

Обозначения моделей разделяются следующим образом. E — исторический correlator Experiment 3, исходный код которого не сохранился. E-R — его воспроизводимая реализация с тем же richest evidence configuration, но без admissibility. F — E-R с описанным pre-ranking gate. F0 — тот же F, замороженный вместе с rules, configuration и scoring до первой held-out инъекции. Поэтому прямой эффект gate оценивается только в паре E-R/F или E-R/F0.

Качество локализации оценивается Top-k, mean reciprocal rank (MRR) и candidate coverage. Для N инъекций.

$$Top - k = (1/N) \cdot \Sigma\, I(r_i \leq k), \quad (5)$$

$$MRR = (1/N) \cdot \Sigma\, (1/r_i), \quad (6)$$

где r_i — ранг правильной причины, а при ее отсутствии в множестве кандидатов вклад в MRR принимается равным нулю. Candidate coverage определяется как доля инъекций, для которых правильная причина вообще присутствует среди кандидатов. Последняя метрика принципиальна: повышение качества ranker не может исправить случай, в котором правильный кандидат не был сформирован.

Экспериментальная методика. Состав четырех экспериментальных этапов и объем обработанных данных приведены в таблице 1.

Таблица 1. Состав экспериментов

Эксперимент	Назначение	Объем данных
1	Пассивная оценка признаков корреляции	149 505 journald; 9000 отношений
2	Углубленная syscall/stack-трассировка	714 842 syscall; 19 719 сохранено
3	3 известных механизма × 5 запусков	231 041 событий; 15 controls
4	4 held-out механизма × 5 запусков	1 043 536 событий; 20 controls

Примечание: составлено авторами по результатам экспериментов.

Пассивная часть исследования выполнялась на двух длительно работающих виртуальных Linux-узлах, далее Host A и Host B. Узлы имели сопоставимую вычислительную конфигурацию и представляли типовую многокомпонентную серверную среду с JVM-приложениями, реляционными СУБД и in-memory хранилищами. Оба узла непрерывно работали несколько недель; синхронизация времени поддерживалась с субмиллисекундной точностью. Во время наблюдения прямое взаимодействие между двумя узлами не зарегистрировано, поэтому корреляция для них выполнялась внутри каждого хоста.

Экспериментальная среда восстановлена по конфигурации и сохраненным артефактам. Experiments 1–2 наблюдали два production-like узла Rocky Linux 9 на x86-64 с OpenJDK 17.0.19+10-LTS, WildFly 40.0.1.Final и PostgreSQL 15.13 в составе наблюдаемого стека. Experiments 3–4 выполнялись на KVM виртуальной машине Rocky Linux 9.8, kernel 5.14.0-687.12.1.el9_8.x86_64, x86-64, 12 vCPU Intel Xeon Gold 6258R 2.70 GHz и 31,1 GiB RAM. Kernel telemetry собиралась BCC 0.35.0 под Python 3.9.25; использовались libbpf 1.5.0 и iproute2 6.17.0. Fault injection выполнялся с iptables 1.8.10, tc/netem и device-mapper delay target; лабораторный DNS-resolver — dnsmasq 2.85. Синхронизация host clocks выполнялась systemd-timesyncd; in-kernel timestamps брались из monotonic clock и один раз за run привязывались к wall-clock time. Изолированная лаборатория Experiments 3–4 не содержала JVM, реляционной СУБД или in-memory store: приложение и dependency были Python-процессами.

В Experiment 1 применялось 1200-секундное активное наблюдение с интервалом 2 с, анализ 24 ч journald и семидневной истории sar. Было обработано 149 505 событий journald и построено 9000 типизированных отношений. Цель состояла в определении реальных join keys и проверке неоднозначности различных идентификаторов.

Последовательность объединения разнородных источников в единый поток evidence показана на рисунке 2.

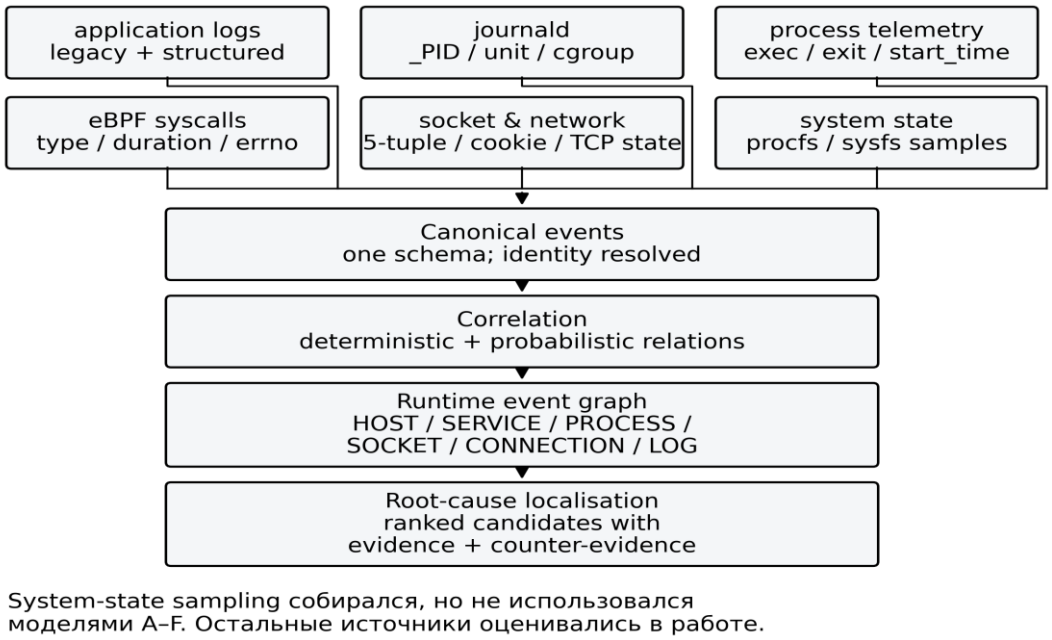


Рисунок 2. Схема формирования единого потока evidence

Примечание – составлено авторами.

Experiment 2 использовал единый eBPF-коллектор и семь последовательных фаз углубленной трассировки. Наблюдалось 714 842 системных вызова, из которых 19 719 были сохранены как события, имеющие диагностическое значение. Дополнительно исследовались user stack, kernel stack и uprobes. До fault-injection была устранена ошибка декодирования AF_INET6 и IPv4-mapped IPv6, обеспечено сохранение destination tuple до очистки полей ядром при неуспешном connect, а также проверен учет жизненного цикла FD.

Контролируемые отказы Experiments 3 и 4 выполнялись в изолированном network namespace fi-lab с veth-парой на узле Host A. Изоляция исключала воздействие tc и iptables на loopback и на исследуемое эксплуатационное ПО. Синтетический клиент использовал connection pool, отдельный database stand-in работал на порту 5432. Для каждого формального запуска использовались стадии PRE 120 с, FAULT 60 с и RECOVERY 120 с.

В Experiment 3 исследовались три класса: F1 — задержка 250±20 мс на сетевом пути к базе данных, F3 — TCP reset/refusal средствами iptables REJECT --reject-with tcp-reset, F10 — принудительное завершение процесса базы с последующим перезапуском. Каждый класс повторялся пять раз. Пилотный SYN-only reject был исключен из основной оценки: пул соединений продолжал использовать уже установленные сокеты и приложение не испытало наблюдаемого эффекта. За 15 формальных запусков получено 231 041 нормализованное событие, потерь ring buffer и очереди не зафиксировано.

Ground truth для каждого класса — fault domain, fault mechanism, команда инъекции и ожидаемые физические эффекты — фиксировался до запуска и хранился отдельно. Коррелятор не открывал ground-truth файлы; они загружались только evaluator после построения predictions. Для оценки detection использовано 15 основных отрицательных окон продолжительностью 60 с, вырезанных из середины PRE-стадий с guard interval 30 с.

Шесть конфигураций A–F различались источниками evidence. A использовала прикладные журналы; B — eBPF, включая системные признаки; C — process/socket/time без syscall attributes; D добавляла syscall type, duration и реальные errno; E объединяла D с семантикой журналов; F получала тот же evidence, что E, но применяла проверку допустимости кандидатов перед ranking. Исторические A–E и новая F измерялись разными реализациями коррелятора, поэтому прямое сравнение F выполнялось только с E-R — переисполненной E в том же воспроизводимом R-correlator.

Experiment 4 предназначался для held-out проверки. До первой формальной инъекции Model F0, правила допустимости, конфигурация и исходный код были зафиксированы и проверялись по SHA-256 в ходе кампании. Новыми механизмами являлись отказ DNS, потеря 15 % пакетов, CPU saturation и disk I/O latency; каждый класс повторялся пять раз, одновременно использовались 20 matched negative controls. Всего получено 1 043 536 нормализованных событий, потерь не зарегистрировано. E-R и F0 оценивались на идентичном потоке данных; изменение правил F0 после начала инъекций не допускалось.

В Experiment 4 единицы анализа разделены заранее. Detection рассчитывается по 40 окнам: 20 fault-windows и 20 matched negative controls. Top-k, MRR, candidate coverage и candidate-set statistics рассчитываются только по 20 fault-windows, поскольку ground-truth cause определена именно для инъекций. Для E-R сумма кандидатов по этим 20 окнам равна 4035, поэтому mean=4035/20=201,75; для F0 сумма равна 19, поэтому mean=19/20=0,95. Именно эти числители и знаменатели используются далее; агрегирование candidate instances по detector-produced incidents вне fault-windows не используется для вывода о качестве локализации, чтобы не смешивать разные единицы анализа.

РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

Информативность межуровневых признаков. Experiment 1 показал, что наиболее устойчивые межуровневые связи формируются не по текстовым именам потоков или по

совпадению временных отметок, а по структурной идентичности. На практике наиболее информативными оказались метаданные journald _PID, _SYSTEMD_UNIT и cgroup, идентичность host+PID+process_start_time, TGID, связь inode/socket с процессом, полный 5-tuple и интервалы жизни сокета. Шаблон журнала и временная близость полезны как контекст, но не являются самостоятельным детерминированным ключом.

Имя потока оказалось небезопасным ключом. На Host A поток, записывавший журнал, и поток, выполнявший сетевую операцию, действительно различались; на Host B поле comm дополнительно усекалось ограничением TASK_COMM_LEN и форматированием, что давало отношение неоднозначности до 1:200. Следовательно, thread name может использоваться как подпись, но не как join key.

Необходимость eBPF для lifecycle событий подтверждается поведением короткоживущих процессов: около 89 % завершались менее чем за одну секунду, а периодическое polling-наблюдение пропускало порядка 85 % таких процессов. Аналогичное ограничение обнаружено для FD. В выборке Host B только 41,2 % снимков /proc/PID/fd однозначно соответствовали одному сокету, а 50,6 % были неоднозначны вследствие быстрого переиспользования дескриптора. Поэтому процесс и сокет в графе должны идентифицироваться с учетом времени жизни.

Experiment 2 позволил отделить диагностически значимые системные события от фонового потока. На Host A 99,6 % отрицательных return values, а на Host B практически 100 % приходились на штатные EAGAIN/EINPROGRESS. Использование условия ret<0 как общего сигнала ошибки формировало бы преимущественно ложную тревогу. После явной классификации errno реальными сетевыми ошибками в наблюдаемом окне оставались, в частности, 58 событий ECONNRESET.

Стековая информация также не дала универсального улучшения. На syscall-exit tracepoint для 9436 событий был получен один и тот же kernel stack, т.е. признак не содержал различительной информации. User stacks были полезны для некоторых native-путей, например libcurl и native database client, но JVM stacks при конфигурации без PreserveFramePointer и perf-map часто завершались на границе JVM/native. Добавление высокочастотных futex/epoll/poll увеличивало стоимость коллектора приблизительно в семь раз, не давая сопоставимого выигрыша корреляции.

Таким образом, сигналы можно разделить на четыре практических класса: детерминированные или сильные структурные (process identity, socket identity, 5-tuple, systemd/cgroup), диагностические атрибуты (реальный errno, syscall type, duration), контекстные/семантические (timestamp, log template, текстовое значение) и небезопасные для связывания (thread name, PID или FD без lifetime, общий признак отрицательного return). Такое разделение использовалось при построении моделей Experiment 3.

Контролируемое внесение отказов и абляция признаков. Наборы evidence, использованные в конфигурациях A–F, сведены в таблице 2.

Таблица 2. Наборы evidence моделей A–F

Модель	Журналы	eBPF / структура	syscall / errno	Admissibility
A	+	–	–	–
B	–	широкий eBPF/system	+	–
C	–	process/socket/time	–	–

Модель	Журналы	eBPF / структура	syscall / errno	Admissibility
D	–	process/socket/time	+ type/duration/real errno	–
E	+	D + доп. system/eBPF	+	–
F	+	как E	+	+

Исторические результаты Experiment 3 показали немонотонную зависимость между объемом evidence и качеством RCA. Модель A, использовавшая только журналы, и наиболее насыщенная модель E имели одинаковое MRR=0,667. При этом среднее число кандидатов составляло соответственно 1,62 и 124,83. Модель B имела лучшее историческое F1 детектора — 0,968, но не локализовала сетевую задержку. Следовательно, способность обнаружить инцидент и способность правильно назвать его причину являются различными задачами.

Особенно показателен F1. Медианная прикладная задержка возросла приблизительно с 5 до 254–256 мс, однако во всех пяти запусках приложение сформировало ноль ошибок. В течение 60-секундного fault-window было зарегистрировано 41 eBPF-событие, в том числе 23 медленных системных вызова, но ни один из них не содержал destination, относящийся к задерживаемой операции. Клиент ожидал внутри poll(), который в постоянной конфигурации агрегировался, а не сохранялся per-event. Поэтому выбранная kernel telemetry не содержала endpoint-attributable признака, достаточного для локализации данного механизма. Это не означает принципиальную невидимость задержки для eBPF; результат относится к выбранному набору сенсоров.

Положение механизма admissibility между генерацией кандидатов и ranking и логика отбраковки показаны на рисунке 3.

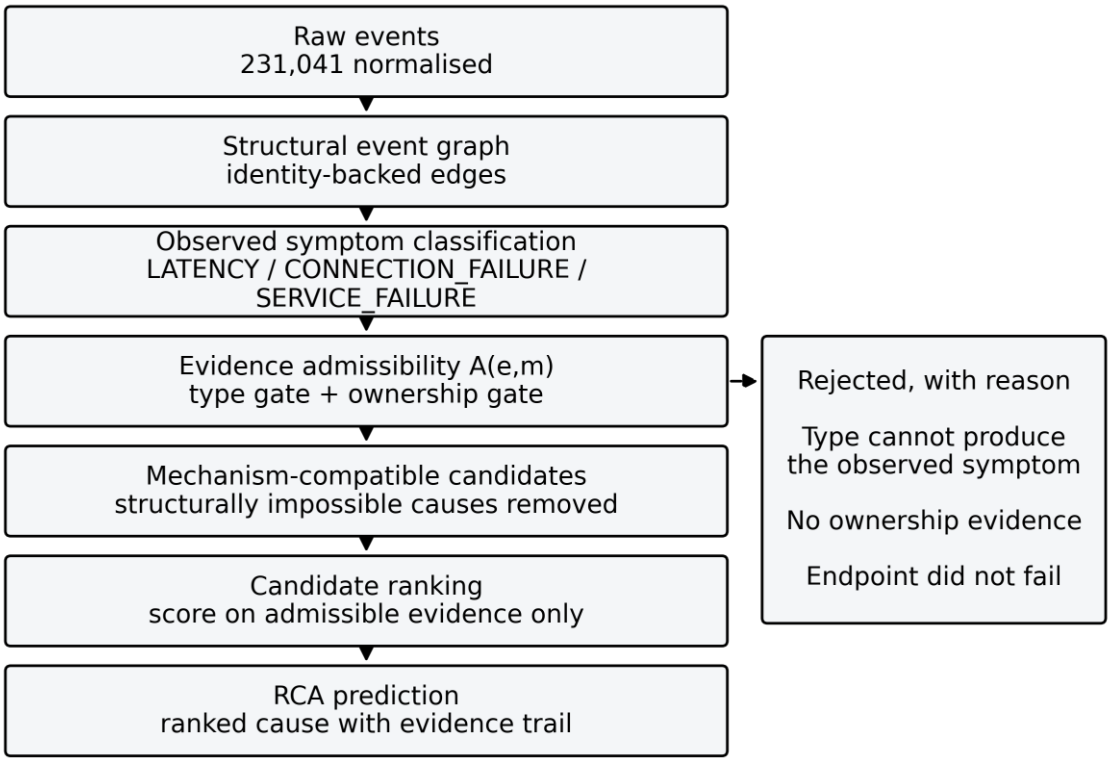


Рисунок 3. Механизм-ориентированная фильтрация причинных кандидатов
Примечание. составлено авторами.

Переход от модели C к D иллюстрирует ценность реального errno и атрибутов системного вызова: исторический Top-1 увеличился с 0,333 до 0,600. При TCP reset именно errno позволял отличить структурно похожее соединение от соединения с подтвержденным механизмом отказа. Одновременно количество кандидатов возрастало, что показывает необходимость разделять информативность признака и объем создаваемых им гипотез.

Для прикладного слоя выполнена отдельная field ablation по четырем структурированным полям: PID, request_id, dst_endpoint и latency_ms. Минимально достаточной комбинацией в исследованной single-hop схеме оказалась {dst_endpoint, latency_ms}: она обеспечивала Top-1=1,000, MRR=1,000 и coverage=1,000. Наличие endpoint без latency давало Top-1=0,667, а конфигурации без endpoint — 0,267. Исключение request_id или PID из полного набора не изменяло результат. Последнее наблюдение относится только к данной одноперeходной схеме и не переносится автоматически на распределенные multi-hop системы.

После re-analysis была реализована модель F с механизм-ориентированной допустимостью. Поскольку исходный исторический correlator Experiment 3 не сохранился, F сравнивалась только с E-R в одном R-correlator. При одинаковом evidence E-R получила Top-1=0,667, Top-3=1,000, MRR=0,833 и медиану 95 кандидатов; F — Top-1=1,000, Top-3=1,000, MRR=1,000 и медиану 2 кандидата при том же coverage=1,000. Этот результат показывает эффект фильтра на известных механизмах, но сам по себе не доказывает обобщение.

Результаты абляции структурированных полей прикладного журнала приведены в таблице 3.

Таблица 3. Абляция структурированных полей журнала

Конфигурация	Top-1	MRR	Coverage
endpoint + latency_ms	1,000	1,000	1,000
endpoint без latency_ms	0,667	0,667	0,667
без endpoint (прочие поля)	0,267	0,267	0,267
full – request_id	1,000	1,000	1,000
full – PID	1,000	1,000	1,000

Примечание: составлено авторами по результатам экспериментов.

Проверка на ранее не использовавшихся механизмах. Для устранения неоднозначности знаменателей сводные показатели Experiment 4 приведены в таблице 4 с явным указанием единицы анализа.

Таблица 4. E-R и F0 на отложенных механизмах

Показатель / единица анализа	E-R	F0
Detection: 20 fault + 20 control windows	TP=17; FP=5; TN=15; FN=3	TP=17; FP=5; TN=15; FN=3
Precision / Recall / F1 / Specificity	17/22=0,773 / 17/20=0,850 / 34/42=0,809 / 15/20=0,750	17/22=0,773 / 17/20=0,850 / 34/42=0,809 / 15/20=0,750
Domain Top-1 / Top-3 / MRR, n=20 fault-windows	9/20=0,450 / 9/20=0,450 / 0,450	9/20=0,450 / 9/20=0,450 / 0,450
Candidate coverage, n=20 fault-windows	9/20=0,450	9/20=0,450
Mechanism Top-1, n=20 fault-windows	0/20=0,000	0/20=0,000
Candidates in fault-windows: total / mean	4035 / 201,75	19 / 0,95
Candidates in fault-windows: median / p95	25 / 668	1 / 1

Примечание. Detection metrics рассчитаны по 20 fault-windows и 20 matched negative controls; Top-k, MRR, coverage и candidate-set statistics — только по 20 fault-windows. Для candidate mean используются точные totals 4035 и 19.

Held-out Experiment 4 был построен специально для проверки последнего ограничения. Model F0, соответствующая F, была заморожена до formal injections. Четыре новых механизма — DNS failure, 15 %-ная packet loss, CPU saturation и disk I/O latency — отсутствовали среди трех механизмов, по которым проектировались правила допустимости.

Для E-R и F0 detection оказался одинаковым: TP=17, FP=5, TN=15, FN=3; precision=0,773, recall=0,850, F1=0,809 и specificity=0,750. По domain-localization обе модели получили одинаковые Top-1=9/20=0,450, Top-3=9/20=0,450 и MRR=0,450. Таблица 5 показывает происхождение этих девяти корректных локализаций: четыре относятся к 15 % packet

loss (4/5), а пять — к disk I/O latency (5/5). DNS failure и CPU saturation не дали корректной domain-localization. Mechanism-level Top-1 остался 0/20 для E-R и F0.

Распределение размера candidate set по конфигурациям для 20 held-out fault-windows представлено на рисунке 4.

Сокращение candidate set воспроизвелось на четко определенной единице анализа — 20 held-out fault-windows. E-R сформировала суммарно 4035 кандидатов (mean=4035/20=201,75; median=25; p95=668), а frozen F0 — 19 кандидатов (mean=19/20=0,95; median=1; p95=1).

Таблица 5. Разбивка результатов Experiment 4 по held-out механизмам

Held-out механизм (n=5)	Detection	Domain Top-1 E-R	Domain Top-1 F0	Верный candidate	Невер-ный set	Пустой set
DNS failure	5/5	0/5	0/5	0/5	5/5	0/5
15 % packet loss	4/5	4/5	4/5	4/5	0/5	1/5
CPU saturation	3/5	0/5	0/5	0/5	3/5	2/5
disk I/O latency	5/5	5/5	5/5	5/5	0/5	0/5
Итого	**17/20**	**9/20**	**9/20**	**9/20**	**8/20**	**3/20**

Примечание: Каждый held-out механизм исследован в пяти независимых запусках. Показатели характеризуют наблюдаемый результат данной экспериментальной кампании; n=5 для каждого механизма и n=20 fault-windows в совокупности недостаточны для надежной оценки устойчивости результатов в иных системах, конфигурациях сенсоров и классах отказов.

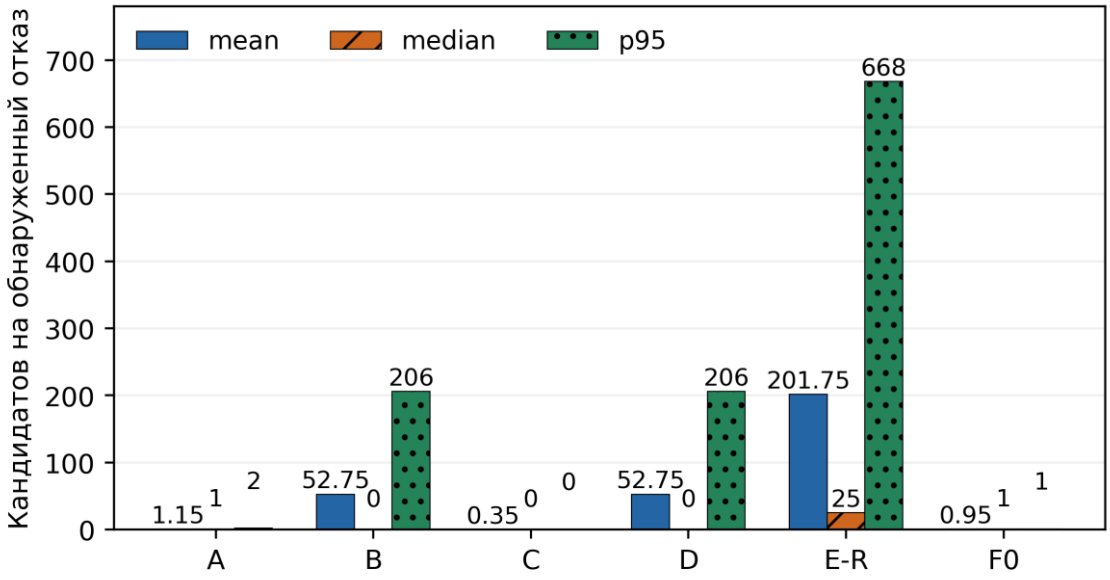


Рисунок 4. Размер множества кандидатов на 20 held-out fault-windows

Примечание – составлено авторами.

При этом candidate coverage для обеих моделей осталось одинаковым: 9/20=0,450. Следовательно, на данной выборке gate резко уменьшал число уже сформированных

гипотез, но не повышал диагностическую полноту и не создавал отсутствующую правильную причину. Не наблюдалось случая, когда F0 удаляла правильный candidate, присутствовавший в E-R на том же fault-window; это наблюдение ограничено данной выборкой и не является доказательством общей безопасности фильтра.

Девять корректных локализаций домена получены только на двух held-out механизмах: 15 % packet loss (4/5) и disk I/O latency (5/5). Неверные candidate set наблюдались во всех пяти DNS-run и в трех обнаруженных CPU saturation run; три пустых candidate set соответствовали одному недетектированному packet-loss run и двум недетектированным CPU-saturation run. Таким образом, CORRECT + WRONG + EMPTY = 9 + 8 + 3 = 20. F0 сохранил то же candidate coverage 9/20, что и E-R, при сокращении числа кандидатов 4035 → 19 (в среднем 201,75 → 0,95 на fault-window). Это наблюдение относится только к данной выборке и не доказывает общей безопасности admissibility gate.

DNS failure выявил ограничение причинного представления. Во всех пяти запусках fault был детектирован, приложение сформировало 25 812 ошибок EAI_AGAIN, а kernel layer — 51 624 UDP send/receive errors в направлении resolver. В контрольных окнах эти признаки отсутствовали. Однако causal vocabulary не содержал типа DNS_DEPENDENCY/NAME_RESOLUTION_FAILURE; в результате алгоритм связывал симптом с database endpoint, несмотря на то что база данных оставалась работоспособной и обслужила 2933 успешных запроса в fault-windows. Здесь evidence существовал, но модель не могла корректно выразить механизм.

Для packet loss наблюдались задержка и деградация передачи, однако постоянный sensor не собирал retransmission, RTO или RTT как endpoint-specific evidence. В результате фиксировались последствия, но отсутствовал прямой признак потери пакетов. Это пример observability boundary. CPU saturation также не сопровождалась scheduler/throttling telemetry, достаточной для уверенного отделения CPU contention от других форм задержки.

Для disk I/O latency ситуация была иной. В fault-windows было зарегистрировано 8474 медленных fsync/fdatasync против 177 в control windows, т.е. системный признак механизма присутствовал. Domain базы данных мог определяться правильно, однако механизм нередко назывался NETWORK_LATENCY_TO. Такой случай показывает representation boundary: наблюдение доступно, но набор причинных типов не позволяет превратить его в правильную гипотезу.

Единицей error analysis является model-run pair: одна модель, примененная к одному injected fault run. В held-out оценке использованы шесть моделей (A, B, C, D, E-R и frozen F0) и 20 fault-runs, поэтому знаменатель равен 6×20=120 model-run pairs. Неуспешной считается пара, в которой ground-truth domain не занял первое место; таких пар было 93/120=77,5 %, по моделям соответственно 12, 20, 19, 20, 11 и 11. Frozen evaluator присвоил этим 93 парам взаимоисключающую первичную категорию: 41 MISSING_EVIDENCE, 46 OBSERVABILITY_BOUNDARY и 6 CANDIDATE_GENERATION_FAILURE; RANKING_FAILURE и MODEL_ADMISSIBILITY_FAILURE получили 0. Mechanism-level ошибки отдельно в этот знаменатель не включались. При этом frozen classifier не выделял REPRESENTATION_BOUNDARY как отдельный класс: например, пять DNS-run F0 ретроспективно удовлетворяют этому определению, но в исходных счетчиках входят в MISSING_EVIDENCE. Поэтому таблица 6 сохраняет исходные counts, а operational taxonomy ниже явно фиксирует различие категорий.

Обсуждение результатов

В исследованных конфигурациях RCA включал наблюдение, нормализацию, структурное связывание, причинное представление, генерацию кандидатов и ранжирование. Основные неуспехи возникали до ранжирования: отсутствовали необходимые

свидетельства или корректный кандидат. Поэтому усложнение ranking само по себе не устраняло наблюдавшиеся ошибки.

Устойчивая идентичность важнее объёма событий. Host+PID+start_time, жизненный цикл сокета и 5-tuple образуют структурный каркас корреляции; массовый сбор событий без этих связей расширяет пространство гипотез.

Таблица 6. Операциональные критерии классификации ошибок RCA

Категория	Критерий присвоения	Пример из Experiment 4
MISSING_EVIDENCE	Нужный признак присутствует в полном записанном потоке, но отсутствует в текущей конфигурации модели из-за исключенного источника или поля. Категория проверяется после исключения observability boundary.	Ablation-конфигурация не использует поле, доступное в полном evidence stream.
OBSERVABILITY_BOUNDARY	Нужный физический признак отсутствует во всем записанном потоке, поскольку активный sensor его не собирал.	Packet loss без endpoint-specific retransmission/RTO/RTT; CPU saturation без scheduler/throttling telemetry.
REPRESENTATION_BOUNDARY	Релевантный evidence присутствует в текущей конфигурации, но causal vocabulary не содержит типа, способного выразить правильный механизм.	DNS: EAI_AGAIN и UDP resolver errors наблюдаются, но DNS_DEPENDENCY/NAME_RESOLUTION_FAILURE отсутствует; disk I/O evidence может быть выражен ошибочным NETWORK_LATENCY_TO.
CANDIDATE_GENERATION_FAILURE	Evidence присутствует и правильный тип причины допускается словарем, но генератор не создает соответствующий candidate.	Корректная сущность/отношение не материализованы в candidate set.
MODEL_ADMISSIBILITY_FAILURE	Правильный candidate существует до gate, но удаляется правилом F/F0.	Проверяется сравнением pre-gate и post-gate candidate sets.
RANKING_FAILURE	Правильный candidate сохраняется после gate, но получает ранг >1.	Проверяется только после успешного прохождения предыдущих стадий.

Примечание. Operational categories применяются последовательно и взаимоисключающе. Числа 41/46/6/0/0 сохранены из frozen evaluator; REPRESENTATION_BOUNDARY в исходном классификаторе отдельным счетчиком не инструментировалась и поэтому ретроспективно описывается, но без повторного запуска классификатора опубликованные counts не перераспределяются.

Историческая модель E иллюстрирует этот эффект: при MRR, равном log-only модели A, число кандидатов было приблизительно в 77 раз больше.

Kernel telemetry связывает процесс, сокет, endpoint и errno, но не всегда определяет ожидающую ресурс логическую операцию приложения. В F1 атрибутируемая задержка отражалась в прикладном журнале. В исследованной single-hop схеме endpoint+latency_ms было достаточно; request_id не давал измеримого выигрыша. Для распределённых систем полезность контекста запроса требует отдельной проверки.

Локализацию fault domain следует отделять от fault mechanism. При disk I/O latency алгоритм определял database dependency, но называл NETWORK_LATENCY_TO. Это направление поиска для оператора, а не корректный диагноз механизма.

На 20 held-out окнах F0 сократила число кандидатов с 4035 до 19 (mean 201,75→0,95) при неизменных domain Top-1 и coverage 9/20=0,450. В этой выборке фильтр уменьшил пространство сформированных гипотез и не удалил доступную правильную причину. Это не доказывает повышения полноты, обобщения или универсальной безопасности фильтра.

В информационной безопасности корреляция может поддерживать triage и incident response при нарушениях доступности. Ссылка normalized event на raw record позволяет проверять evidence и реконструировать события. Метод не устанавливает злонамеренность или attribution: необходимы audit/authentication logs, IDS/NDR/EDR evidence и контекст изменений.

Совершенствование диагностики должно включать сенсоры, устойчивые идентичности, причинный словарь и генерацию кандидатов до усложнения ranking. Обучаемая модель может выбирать среди сформированных гипотез, но не заменяет отсутствующее наблюдение или представление причины.

Для прототипа Qalqan результаты мотивируют сбор структурно сильных признаков: lifecycle процессов и сокетов, 5-tuple, TCP state, errno и распределений задержки. Углублённая трассировка, user stacks и uprobes должны включаться целенаправленно; максимальный объём данных сам по себе не обеспечивает диагностического преимущества.

Ограничения исследования. Экспериментальные результаты имеют несколько ограничений. Пассивные наблюдения выполнены на двух Linux-узлах одного класса эксплуатационных систем. Контролируемые инъекции проводились на одном хосте в изолированной лабораторной среде и включали сначала три, затем четыре класса отказов по пять повторений. Fault-injection application является синтетическим; его структурированный журнал богаче журналов реально наблюдаемых приложений.

Исходный исторический correlator Experiment 3 не был сохранен. Поэтому Model F не сравнивается напрямую с исторической E; корректным является только сравнение F с E-R в одном R-correlator. Исторические и reimplemented metrics в работе рассматриваются раздельно.

Механизм допустимости F0 был заморожен до Experiment 4, однако исходный causal vocabulary проектировался для трех механизмов Experiment 3. Held-out проверка показала именно ограничение такого словаря: ни один из четырех новых fault mechanisms не был назван правильно. Следовательно, результаты не дают основания считать предложенный gate универсальным RCA-методом.

Не исследовались длинные multi-hop цепочки с передачей request context, комбинированные отказы и часть сетевых механизмов. TCP RTT, RTO и retransmission не оценивались как постоянные низкозатратные признаки; JVM stacks исследовались без специальных параметров, улучшающих Java frame unwinding. Пять повторений на класс достаточны для оценки воспроизводимости эффекта, но недостаточны для сильных параметрических статистических выводов, поэтому основное внимание уделено описательным метрикам и повторяемости.

Воспроизводимость также ограничена различием сред Experiments 1–2 и Experiments 3–4: первые наблюдали production-like Java/WildFly/PostgreSQL стек, тогда как fault-injection campaign выполнялась в изолированной Python-based лаборатории. Точные версии основной среды и toolchain приведены выше, но результаты не следует автоматически переносить на другие kernel versions, hypervisors, языковые runtimes или multi-hop topology без отдельной проверки.

ЗАКЛЮЧЕНИЕ

Исследована корреляция журналов, journald и eBPF-телеметрии для локализации отказов Linux-систем. Устойчивые связи обеспечивали идентичность процесса с временем запуска, жизненный цикл сокета, 5-tuple и реальные errno. Имя потока, PID или FD без жизненного цикла и общий признак отрицательного return value не являлись надёжными ключами.

Рост объёма телеметрии не гарантировал улучшения RCA. На известных механизмах фильтрация улучшала локализацию в парном сравнении E-R и F. На четырёх новых классах F0 сократила число кандидатов с 4035 до 19 (mean 201,75→0,95) при неизменных domain Top-1 и coverage 9/20=0,450. Это результат данной выборки, а не доказательство высокой полноты или общей безопасности фильтра.

Выявлены границы наблюдаемости, доступности свидетельств, причинного представления и генерации кандидатов. Корреляция может поддерживать разбор инцидентов критичных сервисов, но не заменяет детекторы атак и контекст для различения злонамеренных и эксплуатационных причин. Перенос результатов на другие среды требует новых экспериментов.

КОНФЛИКТ ИНТЕРЕСОВ. Авторы заявляют об отсутствии финансовых и нефинансовых конфликтов интересов, способных повлиять на результаты или интерпретацию исследования.

ФИНАНСИРОВАНИЕ. Исследование выполнено без внешнего грантового финансирования.

ЭТИЧЕСКОЕ ОДОБРЕНИЕ. Не применимо. Исследование посвящено программно-техническим системам и не включает исследования с участием людей или животных.

ИНФОРМИРОВАННОЕ СОГЛАСИЕ. Не применимо.

ДОСТУПНОСТЬ ДАННЫХ. Агрегированные результаты исследования и описание экспериментальной методики представлены в статье. По обоснованному запросу автор-корреспондент может предоставить материалы воспроизведения, не содержащие эксплуатационных и конфиденциальных данных: обезличенные конфигурации сенсоров и коллектора, правила нормализации и механизм-ориентированной допустимости, конфигурацию R-correlator/F0, сценарии контролируемых fault injections, описание ground-truth schema, а также агрегированные результаты и метрики экспериментальных запусков. Исходные эксплуатационные журналы, полная эксплуатационная телеметрия и сведения, способные раскрыть конфигурацию реальной инфраструктуры, не публикуются по соображениям информационной безопасности.

БЛАГОДАРНОСТИ. Авторы благодарят коллег, участвовавших в технических обсуждениях экспериментальной архитектуры и проверке воспроизводимости процедур сбора телеметрии.

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА. При подготовке рукописи использовался ChatGPT (OpenAI) для языкового и структурного редактирования, перевода аннотации и ключевых слов, а также приведения оформления к требованиям журнала (OpenAI, 2026). Постановка исследования, экспериментальные

данные, программная реализация, численные результаты и их научная интерпретация сформированы и проверены авторами.

СПИСОК ЛИТЕРАТУРЫ

Barata, L. M., Sequeira, S., Lopes, E., Inácio, P. R. M., & Freire, M. M. (2026). Anomaly detection and root-cause identification in microservices: A survey. Cluster Computing, 29, Article 309. https://doi.org/10.1007/s10586-026-06095-9

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1285–1298). ACM. https://doi.org/10.1145/3133956.3134015

Fonseca, R., Porter, G., Katz, R. H., Shenker, S., & Stoica, I. (2007). X-Trace: A pervasive network tracing framework. In Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI) (pp. 271–284). USENIX Association.

He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE International Conference on Web Services (ICWS) (pp. 33–40). IEEE. https://doi.org/10.1109/ICWS.2017.13

Huang, J., Chen, P., Yu, G., Wang, Y., Huang, H., & He, Z. (2024). FaaSRCA: Full lifecycle root cause analysis for serverless applications. In 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE) (pp. 415–426). IEEE. https://doi.org/10.1109/ISSRE62328.2024.00047

Huang, H., Chen, C., Chen, K., Chen, P., Yu, G., He, Z., Wang, Y., Zhang, H., & Zhou, Q. (2025). Mint: Cost-efficient tracing with all requests collection via commonality and variability analysis. In Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (pp. 683–697). ACM. https://doi.org/10.1145/3669940.3707287

Lee, C., Yang, T., Chen, Z., Su, Y., & Lyu, M. R. (2023). Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1750–1762). IEEE. https://doi.org/10.1109/ICSE48619.2023.00150

Liu, D., He, C., Peng, X., Lin, F., Zhang, C., Gong, S., Li, Z., Ou, J., & Wu, Z. (2021). MicroHECL: High-efficient root cause localization in large-scale microservice systems. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 338–347). IEEE. https://doi.org/10.1109/ICSE-SEIP52600.2021.00043

OpenAI. (2026, September 13). ChatGPT (GPT-5.6 Sol): Language and structural editing, translation, and journal-format adaptation of a manuscript on cross-layer Linux/eBPF telemetry correlation [Large language model]. https://chatgpt.com/

Ramachandran, G. S., McDonald, L., & Jurdak, R. (2023). FUSE: Fault diagnosis and suppression with eBPF for microservices. In Service-Oriented Computing (pp. 243–257). Springer. https://doi.org/10.1007/978-3-031-48421-6_17

Satapathy, U., Borse, H., Bachhawat, R., Dalmia, N., Chattopadhyay, S., & Chakraborty, S. (2026). XPLOG: A dynamic observability framework for distributed sandboxed microservices. IEEE Transactions on Services Computing, 19(1), 794–807. https://doi.org/10.1109/TSC.2025.3612900

Sharma, B., & Nadig, D. (2024). eBPF-enhanced complete observability solution for cloud-native microservices. In 2024 IEEE International Conference on Communications (ICC). IEEE. https://doi.org/10.1109/ICC51166.2024.10622329

Wang, H., Wu, Z., Jiang, H., Huang, Y., Wang, J., Köprü, S., & Xie, T. (2021). Groot: An event-graph-based approach for root cause analysis in industrial settings. In 2021 36th IEEE/ACM

International Conference on Automated Software Engineering (ASE) (pp. 419–429). IEEE. https://doi.org/10.1109/ASE51524.2021.9678708

Wu, L., Tordsson, J., Elmroth, E., & Kao, O. (2020). MicroRCA: Root cause localization of performance issues in microservices. In 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS) (pp. 1–9). IEEE. https://doi.org/10.1109/NOMS47738.2020.9110353

Xie, Z., Zhang, S., Geng, Y., Zhang, Y., Ma, M., Nie, X., Yao, Z., Xu, L., Sun, Y., Li, W., & Pei, D. (2024). Microservice root cause analysis with limited observability through intervention recognition in the latent space. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 6049–6060). ACM. https://doi.org/10.1145/3637528.3671530

Yu, G., Chen, P., Li, Y., Chen, H., Li, X., & Zheng, Z. (2023). Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 553–565). ACM. https://doi.org/10.1145/3611643.3616249

Zhang, Y., Guan, Z., Qian, H., Xu, L., Liu, H., Wen, Q., Sun, L., Jiang, J., Fan, L., & Ke, M. (2021). CloudRCA: A root cause analysis framework for cloud computing platforms. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (pp. 4373–4382). ACM. https://doi.org/10.1145/3459637.3481903

Zhang, S., Jin, P., Lin, Z., Sun, Y., Zhang, B., Xia, S., Li, Z., Zhong, Z., Ma, M., Jin, W., Zhang, D., Zhu, Z., & Pei, D. (2023). Robust failure diagnosis of microservice system through multimodal data. IEEE Transactions on Services Computing, 16(6), 3851–3864. https://doi.org/10.1109/TSC.2023.3290018

Zhao, Z., Wang, Z., Zhang, T., Shen, Z., Dong, H., Lei, Z., Ma, X., Xu, G., Ding, Z., & Yang, Y. (2026). CHASE: A causal hypergraph based framework for root cause analysis in multimodal microservice systems. IEEE Transactions on Big Data. Advance online publication. https://doi.org/10.1109/TBDATA.2026.3725931

Zheng, L., Chen, Z., He, J., & Chen, H. (2024). MULAN: Multi-modal causal structure learning and root cause analysis for microservice systems. In Proceedings of the ACM Web Conference 2024 (pp. 4107–4116). ACM. https://doi.org/10.1145/3589334.3645442

Zhu, Z., Lee, C., Tang, X., & He, P. (2024). HeMiRCA: Fine-grained root cause analysis for microservices with heterogeneous data sources. ACM Transactions on Software Engineering and Methodology, 33(8), Article 200, 1–25. https://doi.org/10.1145/3674726

Авторлар туралы мәліметтер
Информация об авторах
Information about authors



Арефьев Александр Владимирович– физика-математика ғылымдарының кандидаты, доцент, Санкт-Петербург мемлекеттік аэроғарыштық аспап жасау университеті, Санкт-Петербург, Ресей.
Арефьев Александр Владимирович– кандидат физико-математических наук, доцент, Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия.
Arefyev Alexander Vladimirovich - Candidate of Physical and Mathematical Sciences, Associate Professor, Saint Petersburg State University of Aerospace Instrumentation, Saint Petersburg, Russia.
E-mail: aaref@guap.ru,
ORCID: 0000-0002-2860-790X



Фарафонов Виктор Георгиевич - физика-математика ғылымдарының докторы, профессор, Санкт-Петербург мемлекеттік аэроғарыштық аспап жасау университеті, Санкт-Петербург, Ресей.
Фарафонов Виктор Георгиевич– доктор физико-математических наук, профессор, заведующий кафедрой прикладной математики Санкт-Петербургского государственного университета аэрокосмического приборостроения (ГУАП), Санкт-Петербург, Россия.
Farafonov Viktor Georgievich– Doctor of Physical and Mathematical Sciences, Professor, Saint Petersburg State University of Aerospace Instrumentation, Saint Petersburg, Russia.
E-mail: far@guap.ru,
ORCID: 0009-0005-6009-1543.



Тулегенов Арман Рашитович – компьютерлік қауіпсіздік саласындағы магистр, тәуелсіз зерттеуші, «МАЭК» ЖШС, Астана, Қазақстан.
Тулегенов Арман Рашитович – магистр наук в области компьютерной безопасности, независимый исследователь, ТОО «МАЭК», Астана, Казахстан.
Tulegenov Arman Rashitovich – M.Sc. in Computer Security, Independent Researcher, MAEK LLP, Astana, Kazakhstan.
E-mail: kaz-arman-tulegenov@yandex.ru,
ORCID: 0009-0000-0764-4771.