Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

eBPF для сисадмина: мониторинг и безопасность Linux-серверов

Разворачиваем bcc/bpftrace, ebpf_exporter и Tetragon на боевых серверах — без перезагрузок

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Linux-серверы клиента — 1С с PostgreSQL, файловые, Docker-хосты — «тормозят», а top и iostat причину не видят; strace замедляет процесс в разы, tcpdump заливает диск. Диагностика тянется днями и кончается перезагрузкой «на всякий случай». Мы закрываем это eBPF-стеком: трейсинг ядра с overhead в единицы процентов, метрики в Prometheus и контроль запусков процессов.

Почему это важно бизнесу

- Час простоя сервера 1С/БД — остановка счетов и отгрузок; скорость диагностики напрямую сокращает простой
- Диагностика «вслепую» кончается перезагрузкой: проблема возвращается, причина остаётся ненайденной
- strace и tcpdump на проде сами создают инцидент: кратное замедление процесса и переполненный диск
- Посторонний процесс на сервере виден в момент запуска, а не постфактум из логов — короче окно атаки

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

5.10+

ядро с BTF/CO-RE — наш порог допуска сервера к eBPF-стеку (ring buffer и CAP_BPF есть с 5.8; D...

kernel.org / наш стандарт

0.26

актуальный bpftrace; до старта проверяем /sys/kernel/btf/vmlinux — без BTF CO-RE-зонды не собр...

по докам bpftrace.org

≤3%

допустимый overhead наблюдения на прод-узле; выше — переписываем зонд на агрегацию в BPF-map

наш стандарт

50 мс

порог «медленного» fsync в дисковом зонде — всё, что дольше, логируем с PID и командой

наш стандарт

1.7

Tetragon standalone: контроль exec и доступа к файлам; первые 7 дней — только audit-режим

по докам tetragon.io

9435

порт ebpf_exporter v2 для Prometheus: гистограммы задержек диска и сети без демонов-парсеров

README ebpf_exporter v2 (github.com/cloudflare)

РЕАЛИЗАЦИЯ 1 ИЗ 3

Трейсинг-набор на сервере 1С + PostgreSQL

Что настраиваем

Debian 12 / Ubuntu 24.04: серверы СУБД и приложений — первый слой диагностики без агентов и перезагрузок

Как мы это делаем

1. Проверяем базу: uname -r (≥5.10) и наличие /sys/kernel/btf/vmlinux — без BTF CO-RE-зонды не стартуют, ядро меняем на дистрибутивное
2. Ставим apt install bpfcc-tools bpftrace; у дежурного шпаргалка: biolatency-bpfcc, execsnoop-bpfcc, tcpretrans-bpfcc, opensnoop-bpfcc
3. Кладём наши bpftrace-скрипты в /opt/itf-ebpf: медленный fsync (>50 мс), гистограмма vfs_read, задержка getaddrinfo через uprobe на libc
4. Каждый зонд — с фильтром /comm == "postgres"/ или по PID и под timeout(1); бессрочный трейс на проде не оставляем

РЕЗУЛЬТАТ

Причину «тормозов» видим за минуты: гистограмма biolatency сразу разделяет проблему диска и проблему запросов, tcpretrans ловит ретрансмиссии и кривой MTU. Ни одного strace на боевом процессе и ни одной «профилактической» перезагрузки.

КЛЮЧЕВОЙ НЮАНС

Сверяем точки привязки со стабильностью: tracepoint:syscalls переживают обновления ядра, kprobe на внутренние функции ломаются при апдейте — их допускаем только в короткоживущих ad-hoc зондах.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Метрики eBPF в Prometheus и Grafana

Что настраиваем

Узлы постоянного наблюдения: СУБД, файловый сервер, шлюз — ebpf_exporter v2 как systemd-служба

Как мы это делаем

1. Разворачиваем ebpf_exporter v2 (libbpf + CO-RE) бинарником; systemd-unit с DynamicUser и AmbientCapabilities=CAP_BPF CAP_PERFMON вместо работы от root
2. Подключаем готовые конфиги из examples: гистограмма задержек блочного I/O (biolatency) и переполнение TCP SYN backlog (tcp-syn-backlog)
3. Prometheus скрейпит :9435 каждые 15 с; в Grafana — heatmap задержек диска по каждому узлу
4. Алерты: p99 задержки I/O выше 20 мс на NVMe 5 минут подряд и рост ретрансмиссий — в Telegram дежурному

РЕЗУЛЬТАТ

Вместо ad-hoc запусков — исторические гистограммы: деградацию RAID-кэша или рост очереди диска видим по тренду до жалоб пользователей, а на разборе инцидента есть картина «как было за неделю до».

КЛЮЧЕВОЙ НЮАНС

В метрики отдаём только агрегаты из BPF-map (гистограммы, счётчики). Поток «событие-в-строку» через exporter не гоняем: кардинальность Prometheus и overhead растут мгновенно.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Контроль запусков и файлов: Tetragon

Что настраиваем

Docker-хосты и серверы с веб-приложениями — Tetragon 1.7 standalone (без Kubernetes), события в SIEM

Как мы это делаем

1. Ставим Tetragon как systemd-службу/контейнер: из коробки пишет process_exec/process_exit с деревом предков в /var/log/tetragon/tetragon.log
2. Добавляем TracingPolicy: kprobe security_file_permission на /etc/shadow и каталог бэкапов, whitelist легитимных бинарей через matchBinaries
3. Неделя audit-режима: разбираем JSON-события, дополняем whitelist; только затем включаем enforcement (action Sigkill) на явные нарушения
4. События через fluent-bit уходят в Wazuh; отдельный алерт — интерактивный shell из-под пользователя веб-приложения

РЕЗУЛЬТАТ

Запуск постороннего бинарника или чтение /etc/shadow фиксируется в момент события — с PID, путём и родительской цепочкой процессов. Реакция идёт по факту, а не при разборе логов через сутки.

КЛЮЧЕВОЙ НЮАНС

Enforcement — только после audit-периода: Sigkill-политика без обкатки убивает легитимные cron- и backup-задачи. Синтаксис правил сверяем с разделом TracingPolicy документации tetragon.io.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Зонд без фильтра процесса**
  bpftrace без /comm/ или /pid/ обрабатывает все события системы — на нагруженном узле это ощутимый overhead; фильтр в прод-зонде обязателен.
- ✕ **Кастомное ядро без BTF**
  Без CONFIG_DEBUG_INFO_BTF=y нет /sys/kernel/btf/vmlinux — CO-RE-инструменты не стартуют; ставим дистрибутивное ядро или пакет с BTF.
- ✕ **kprobe на внутренние функции**
  Имена функций ядра меняются между версиями — зонд молча отваливается после апдейта; база мониторинга только на стабильных tracepoint.
- ✕ **strace на боевом процессе**
  ptrace-остановка на каждом syscall замедляет процесс в разы; заменяем на execsnoop/syscount с overhead в доли процента.
- ✕ **Бессрочный трейс в фоне**
  BPF-map забытого зонда живут в памяти ядра; каждый ad-hoc запуск — через timeout(1) или systemd-run с лимитами по времени.
- ✕ **Enforcement с первого дня**
  Sigkill-политика Tetragon без audit-обкатки убивает легитимные cron-задачи и бэкапы; сначала неделя наблюдения и whitelist.
- ✕ **Устаревшие пакеты дистрибутива**
  bpftrace/bcc в репозиториях отстают на годы; при нехватке синтаксиса берём upstream-релиз и фиксируем версию в Ansible-роли.
- ✕ **CAP_BPF контейнерам приложений**
  Право загрузки eBPF-программ — почти доступ к ядру; зонды запускаем только с хоста, привилегированные контейнеры не раздаём.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Ядро ≥5.10 с BTF (/sys/kernel/btf/vmlinux) на всех Linux-серверах
- bpfcc-tools + bpftrace установлены, шпаргалка топ-5 инструментов у дежурного
- Регламент: strace/tcpdump на проде — только по согласованию, есть eBPF-замена

НОРМАЛЬНО

- Скрипты /opt/itf-ebpf раскатаны Ansible: fsync, biolatency, tcpretrans с фильтрами
- ebpf_exporter v2 на ключевых узлах, heatmap задержек диска в Grafana
- Алерты по p99 задержек I/O и TCP-ретрансмиссиям в Telegram дежурному
- Журнал запусков процессов: execsnoop-лог или exec-события Tetragon

ХОРОШО

- Tetragon 1.7 с TracingPolicy на критичных хостах, экспорт событий в SIEM
- Точечный enforcement: /etc/shadow, каталоги бэкапов, shell из-под веб-юзера
- Проверка всех eBPF-зондов после каждого обновления ядра — пунктом чек-листа
- Ежеквартальная ревизия политик, whitelist и версий bpftrace/Tetragon

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] На всех Linux-серверах ядро с BTF — файл /sys/kernel/btf/vmlinux существует?
- [ ] Дежурный может за 5 минут снять гистограмму задержек диска (biolatency) на боевом сервере?
- [ ] Медленные fsync и записи логируются с PID и именем процесса, а не ищутся постфактум?
- [ ] Задержки диска и TCP-ретрансмиссии видны в Grafana за последние 30 дней?
- [ ] Запуск нового процесса на сервере СУБД оставляет след с родительской цепочкой?
- [ ] Политики Tetragon прошли audit-период до включения enforcement?
- [ ] После обновления ядра проверяется работоспособность всех eBPF-зондов?
- [ ] strace на боевых процессах запрещён регламентом, замена на eBPF-инструменты описана?
- [ ] Ad-hoc трейсы ограничены по времени (timeout/systemd-run), забытых фоновых зондов нет?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Аудит Linux-парка: версии ядра, BTF, готовность к eBPF-стеку — с планом обновлений
- Разворачиваем трейсинг-набор bcc/bpftrace и скрипты /opt/itf-ebpf через Ansible-роль
- Настраиваем ebpf_exporter → Prometheus/Grafana с алертами в Telegram дежурному
- Внедряем Tetragon по схеме audit → whitelist → точечный enforcement, события в SIEM
- Берём серверы на поддержку: диагностика инцидентов по eBPF-данным, SLA на реакцию

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 bpftrace Manual и One-Liner Tutorial (bpftrace.org — 0.26)

02 BCC Tools Reference Guide (iovisor/bcc) (github.com — 2026)

03 Tetragon: TracingPolicy и Enforcement (tetragon.io — 1.7)

04 ebpf_exporter: README и examples (github.com — v2.x)

05 BPF Documentation: verifier, BTF, maps (docs.kernel.org — 6.x)

06 Наш шаблон Ansible-роли itf-ebpf (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.