

SecureExec — Linux EDR с AI-аналитиком

Технический whitepaper

SecureExec

Содержание

1. В одном абзаце	1
2. Для кого это и какую задачу решает	2
3. Архитектура в общих чертах	2
4. Как устроен Linux-агент	2
5. Детекция	4
6. Реагирование	8
7. UI расследований	9
8. Как установить	11
9. Безопасность и границы доверия	12
10. Что уже работает и что в roadmap	13
11. Коммерческое предложение	13
12. Как начать	14
13. Контакты	14

Технический обзор платформы SecureExec: что это, какую задачу решает, как устроен Linux-агент, как реализованы детекция и реагирование, и как AI-аналитик расследует алерты в реальном времени.

Аудитория: инженеры по безопасности, платформенные и SRE-команды, выбирающие Linux EDR. Статус: май 2026 — соответствует текущей кодовой базе. Пункты roadmap отмечены отдельно.

1. В одном абзаце

SecureExec — это Linux EDR со встроенным AI-аналитиком, поставляемый по подписке. Платформа наблюдает за всеми вашими Linux-серверами на уровне ядра через eBPF, в реальном времени детектирует подозрительное поведение, даёт реагировать из одной консоли и приносит LLM-аналитика, который расследует каждый алерт так, как это сделал бы аналитик L1 — только за секунды, целиком и по каждому алерту.

Вы выбираете, где живёт control plane:

- **Управляемое облако (secureexec.ru).** Регистрируетесь, запускаете один curl на хосте — через 5 минут в консоли уже алерты по живой телеметрии. Без CAPEX, без DevOps, списывается как OPEX по подписке на хост.
- **Self-hosted в вашем контуре.** Тот же продукт, разворачивается в вашем стеке через Docker Compose (single-node) или Kubernetes (production-кластер). Air-gapped поддерживается. Локальная LLM (Ollama, vLLM) оставляет всё внутри периметра.

Возможности продукта одинаковы — различается только то, где работает control plane.

2. Для кого это и какую задачу решает

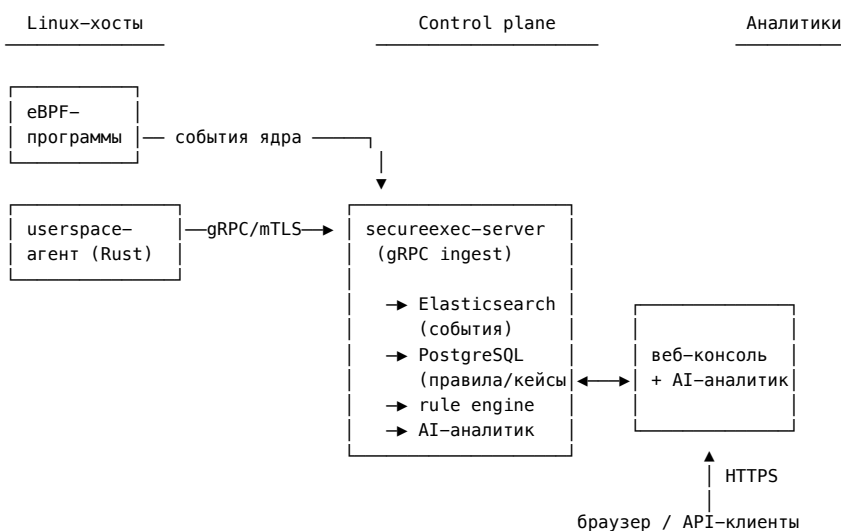
Linux стал production-операционной системой по умолчанию — около **45% серверного рынка и половина облачных нагрузок** работают на нём. Средства защиты для Linux за этим не успевают:

- **Корпоративные EDR** (CrowdStrike, SentinelOne, Defender) на бумаге покрывают Linux, но они cloud-first, closed-source и стоят как корпоративная лицензия — рассчитаны на компании, у которых уже есть полноценная SOC-команда 24/7. В регуляторных, суверенных и air-gapped средах они не приживаются — и не заменяют ту самую SOC-команду, которую вы тоже оплачиваете.
- **Open-source стеки** (Wazuh, Falco + Tetragon) дают отличные сенсоры, но к ним нет единой консоли, AI-аналитика и встроенного реагирования. В итоге вы собираете из 3–4 инструментов собственный pipeline и всё равно нанимаете аналитиков, чтобы разбирать получаемые алерты.

SecureExec сделан для **mid-market и регулируемых команд**, которым нужен настоящий EDR на Linux, но не подходит мультитенантный SaaS и нет ресурсов держать SOC L1 24/7 для первичного разбора алертов. AI-аналитик снижает нагрузку на L1/SecOps и автоматизирует первичный разбор; подписочная модель заменяет CAPEX-проект по сборке такой линии в одиночку.

3. Архитектура в общих чертах

Три компонента. На production-хостах работает только агент; всё остальное живёт в одном месте — на одной VM, в Kubernetes или разнесено по нескольким нодам.



Компоненты:

1. **Linux-агент** — один статически скомпилированный musl-бинарь (~10–15 МБ). Загружает eBPF-программы, отправляет типизированные события на сервер по gRPC/mTLS и выполняет команды реагирования, которые приходят со стороны сервера.
2. **Сервер** — gRPC-приём событий, движок правил, оркестрация AI-агента, диспетчер реагирования. События лежат в Elasticsearch, конфигурация и кейсы — в PostgreSQL.
3. **Веб-консоль + API** — UI для аналитиков; HTTP API и webhooks для интеграций (Slack, Telegram, Email, threat intel).

4. Как устроен Linux-агент

У агента две половины: **eBPF-программы, работающие в ядре**, и **userspace-консьюмер**, который читает события из kernel ring buffers, обогащает их и стримит на сервер.

4.1 eBPF-сенсоры

Агент привязывает eBPF-программы к kernel tracepoints, kprobe и LSM-хукам. Свой kernel-модуль для телеметрии не нужен — агент работает на любом mainline-ядре Linux 5.10+ с BTF. Покрытие:

- **Жизненный цикл процессов** — exec, fork, exit, полный argv и envp, cwd, UID/GID, capabilities.
- **Файловая активность** — openat, creat, unlink, rename, chmod, chown, создание symlink и hardlink, truncate.
- **Сеть** — TCP connect / accept / bind, UDP send, DNS-запросы (полное имя + тип записи), listen-сокеты.
- **Системные вызовы, важные для безопасности** — ptrace, process_vm_readv / writev, memfd_create, загрузка bpf()-программ, capset, setuid / setgid, смены namespace (unshare, sets), загрузка kernel-модулей, keyctl, io_uring_setup, mount / umount, сигналы (kill), отображения памяти с W+X.

В каждом событии:

- Полный **контекст процесса** — PID, PPID, process_guid, путь к exe, exe_hash (SHA-256 бинаря при первом наблюдении), exe_size, hostname, версия ядра.
- **Контейнерный контекст** — container id из cgroup, плюс Kubernetes-лейблы, если есть (pod, namespace, workload).
- **Wall-clock + boot-time timestamps** и идентификатор capture-point eBPF.

4.2 Ring buffers и backpressure

Объём реальный — нагруженный веб-хост на простое генерит 200–500 event/s и до 5k/s в моменты exec-storm. Агент к этому готов:

- Отдельные eBPF ring buffers под каждое семейство событий, размеры по реальному трафику: **16 МиБ** на процессы, **16 МиБ** на файлы, **8 МиБ** на security, **1 МиБ** на сеть.
- В userspace — **epoll-polling**: CPU простаивает, когда ядро не выдаёт событий, и мгновенно просыпается на первое же.
- **Неблокирующий try_send** в 64K-slot mpsc-канал держит ring buffer пустым, даже если downstream-пайплайн на мгновение тормозит. Backpressure всплывает как счётчик потерянных событий в heartbeat, а не как застрявший kernel-ring.

4.3 Транспорт

- Постоянный **gRPC/mTLS**-стрим на каждого агента. Одно соединение, двунаправленное: события идут вверх, команды (kill, isolate, block, live response) — вниз.
- **У каждого агента уникальный клиентский сертификат**, выдаётся при провижининге. Отзыв сертификата мгновенный; агент, потерявший доверие, отваливается на TLS-уровне.
- Агенты **не подтягивают код в рантайме**. Единственный исходящий egress — endpoint control plane на TCP 50051.

4.4 Самообновление агента

Агент поддерживает **самообновление с SHA-256-верификацией**, запускаемое из консоли. Опциональный kernel-модуль (см. §6.1) поставляется через DKMS (deb/rpm) и обновляется условно — при несовпадении SHA весь апдейт отменяется, ABI-skew между агентом и kmod становится невозможным.

4.5 Потребление ресурсов

Измерения на парке 20k+ Linux endpoints (предыдущий проект) с аналогичным eBPF-агентом того же инженера:

Метрика	Типично	p99
RSS агента	40–80 МБ	150 МБ
CPU (простой)	< 0.5 % vCPU	2 %
CPU (всплески exec/fork)	1–3 %	8 %
Поток событий с ноды	50–500 evt/s	5k evt/s
Трафик к серверу	~10–50 КБ/с	500 КБ/с

5. Детекция

5.1 Встроенные правила (63 на сегодня)

Серверный движок правил поставляется с **63 курируемыми детекциями**, сгруппированными по тактикам MITRE ATT&CK. Каждое правило data-driven против типизированной схемы событий, имеет positive/negative unit-тесты и allowlist процессов для снижения FP.

Initial Access / Execution / Persistence (TA0003)

- Reverse shell (классический + fileless), interactive shell over socket
- SSH brute-force и успешный логин после серии неудач
- Криптомайнеры, C2 direct-IP egress, /tmp ехес-дропперы
- Cron, systemd (service/timer/path/socket/mount), udev, at-job, authorized_keys, LD_PRELOAD, file capabilities
- **Новые (май 2026):** shell-init persistence (/etc/profile.d/*, /etc/bash.bashrc), tampering PAM, sshd_config, NSS/DNS (/etc/nsswitch.conf, /etc/hosts), tampering репозитории пакетного менеджера, boot/EFI/initramfs config, закрепление через init-скрипты (/etc/rc.local, rc.d, init.d, update-motd.d, хуки диспетчеров NetworkManager/networkd)

Defense Evasion (TA0005)

- Отключение auditd/SELinux, загрузка kernel-модулей
- **Новые:** очистка shell history, tampering login-логов (utmp/wtmp/btmp), tampering syslog, tampering правил auditd, сброс/отключение firewall (iptables/nft/ufw), злоупотребление chattr +i, self-defense агента (SIGKILL/SIGTERM против агента SecureExec)

Privilege Escalation (TA0004)

- Privilege escalation (с allowlist для container runtimes и priv-sep бинарей), подозрительный sudo, tampering sudoers, sudo brute-force
- **Новые:** PwnKit / CVE-2021-4034 (pkexec с пустым argv), SUID/SGID в user-writable каталогах, GTFOBins SUID abuse, изменение привилегированной группы / учётной записи (usermod -aG sudo, создание учётной записи с UID 0)

Discovery / Lateral Movement (TA0007 / TA0008)

- Namespace escape, pivot_root escape, outbound-аномалии
- **Новые:** web shell execution (web-сервер → shell/интерпретатор), доступ к cloud IMDS (169.254.169.254), reverse SSH tunnel (ssh -R / -D), исходящий port scan (≥50 портов / /24 / 30 с)

Credential Access / Collection

- Чтение credential-файлов, чтение памяти ядра, tampering чувствительных файлов
- Process injection (ptrace, process_vm_wrotev), fileless execution

Impact / Exfiltration (TA0010 / TA0040)

- Ransomware-подобная активность, tampering системных бинарников
- **Новые:** DNS tunneling (длинные QNAME / высокоэнтропийные burst), bulk archive-then-egress (архив в /tmp/ + исходящий connect), disk wipe/format (dd, shred, mkfs.*), mass log destruction (≥5 удалений в /var/log/ за 10 с)

Kernel / Container / Masquerade

- Kernel tamper, снижение hardening ядра, raw socket sniffer
- Process name masquerade, container namespace escape
- Self-monitoring: алармы по дропам и offset-mismatches в eBPF

Правила — **data-driven**, матчатся против типизированной схемы событий, а не over free-form log index. Это делает их детерминированными, быстрыми и удобными для unit-тестов (на каждое правило есть positive и negative тест).

5.2 Кастомные правила и suppression

- Кастомные правила на организацию — через консоль.
- Suppression-правила для глушения известных шумных паттернов (например, бэкап, триггерящий правила tamper) без потери исходных событий.

5.3 AI-аналитик

Автономный LLM-аналитик, который расследует алерты и кейсы так, как это сделал бы человек. Это центральная функция продукта и причина, по которой L1/SecOps не приходится вручную разбирать каждый алерт в три часа ночи.

Двухфазный pipeline расследования gather → synthesize, по дизайну:

1. **Фаза gather.** LLM получает типизированный каталог тулзов и лимит итераций. Он рассуждает о том, что нужно узнать, делает tool-calls (часто параллельными пачками) и собирает структурный FindingsLedger — клеймы с ссылками на event_id, сырые evidence-снимпы, IOCs, отброшенные гипотезы и открытые вопросы.
2. **Фаза synthesize.** Второй LLM-вызов получает исходный бриф и только сам ledger — **никаких тулзов**. Возвращает финальный JSON-вердикт: {verdict, confidence, summary, impact_assessment, recommended_actions, iocs}.

Сплит держит «горячий» цикл рассуждений отдельно от строго-схемного шага вынесения вердикта, не даёт контексту распухнуть на длинных расследованиях и оставляет за каждым вердиктом аудит-trail evidence, который может перепроверить человек.

Каталог тулзов (read-only, типизированы под схему eBPF-событий). Каталог отдан LLM через OpenAI function-calling schema:

- **Тулзы расследования** — get_alert_details, get_case_details, get_process_tree, search_events, get_endpoint_info, get_alert_history.
- **Live response на хосте** (14 read-only тулзов, идущих через live-response канал агента) — host_list_files, host_read_file, host_find_files, host_search_files, host_stat_file, host_hash_file, host_list_processes, host_list_open_files (lsof -p), host_list_sockets (ss), host_list_modules (lsmod), host_show_network, host_login_history (last), host_read_journal (journalctl с фильтрами по времени, unit, приоритету и kernel-only).

Live-response тулзы **опциональны на уровне организации** (off / chat-only / full investigations) и выполняются через тот же аутентифицированный, аудит-логируемый канал, что и ручные действия реагирования из консоли.

Multi-turn чат в кейсе. Аналитик может открыть chat-панель у любого кейса и вести многошаговый диалог с AI-аналитиком — задавать уточняющие вопросы, давать ground-truth («этот аккаунт принадлежит IT, не атакующему»), просить пересмотреть вердикт с учётом нового контекста. Переписка хранится с кейсом и подаётся в последующие расследования как контекст высокой важности: поправки SME перевешивают generic-pattern рассуждения.

Фазы по roadmap.

- **Phase 0 — Copilot (доступен сейчас).** Аналитик жмёт «Расследовать через AI» на алерте или кейсе; двухфазный pipeline обрабатывает и пишет структурированный отчёт в кейс.
- **Phase 1 — Auto-triage (Q3 2026).** Каждый новый алерт выше порога severity расследуется автоматически; уверенные FP суппрессируются, TP уезжают в очередь на человека.
- **Phase 2 — Semi-autonomous response (Q4 2026).** AI предлагает конкретные действия реагирования (kill, isolate, block hash); auto-approve политики исполняют безопасные сами, остальное утверждает человек.
- **Phase 3 — Autonomous hunting (2027).** Постоянный поведенческий контекст на каждый endpoint, проактивные fleet-wide hunt'ы по ledger предыдущих расследований.

LLM endpoint — одна переменная конфигурации на организацию. Можно указать **Amazon Bedrock, Azure AI, локальный vLLM/Ollama**, либо (opt-in) внешний провайдер: OpenAI, Anthropic, OpenRouter. В управляемом облаке LLM включён в подписку.

AI Security Analysis True Positive 95% confidence deepseek/deepseek-v3.2

Summary
Пользователь root на хосте secureexec1 выполнил вредоносный скрипт импланта (/tmp.secureexec-demo/implant.sh) через задание cron, который выполняет beaconing на внешний IP-адрес 198.51.100.1 и сканирование TCP-портов (22, 25, 3389). Это соответствует MITRE ATT&CK T1053.003 (Scheduled Task/Job: Cron) для persistence и T1046 (Network Service Discovery) для разведки.

Impact Assessment
Хост был скомпрометирован с установленным persistence через cron jobs. Злоумышленник имеет постоянный доступ к системе, выполняет разведку сети и поддерживает связь с командным сервером. Это представляет высокий риск для конфиденциальности, целостности и доступности системы.

Recommended Actions

- urgent **isolate host** `secureexec1`
Хост активно скомпрометирован с установленным persistence и выполняет beaconing на внешний командный сервер. Необходимо немедленно изолировать для предотвращения дальнейшего распространения и утечки данных.
- urgent **kill tree** `4f6d0f12962d6ee5630037f9f197069da00e79703110c102b5637e6341c9bfb2`
Остановить все процессы, связанные с вредоносным скриптом импланта и его дочерними процессами для прекращения активности C2.
- high **investigate further** Файлы в `/tmp.secureexec-demo/` и записи `cron`
Необходимо проанализировать содержимое скрипта импланта, проверить другие задания cron и определить полный масштаб компрометации.
- high **block hash** `5b19c58ab9ba3d07e71d2958a225c5bcddf9b5c1`
Заблокировать хэш вредоносного процесса curl, используемого для beaconing, чтобы предотвратить его повторное выполнение.

[▶ 7 tool calls](#)

Рис. 1: AI-аналитик — расследование true-positive: вердикт, уверенность, рекомендованные действия реагирования и полный аудит-trail вызовов тулзов.

AI Security Analysis False Positive 85% confidence deepseek/deepseek-v3.2

Summary
 The alert triggered on a CLONE_NEWPID namespace change via setns syscall, but investigation reveals this is part of legitimate Docker container execution. The process tree shows runc init creating container processes with proper namespace isolation as part of normal container orchestration on host 'sitechecker-us', which runs containerized workloads.

Impact Assessment
 No impact - this is expected behavior for container runtime operations. The namespace changes are part of proper container isolation mechanisms rather than escape attempts.

Recommended Actions

low **review access** sitechecker-us
 Review container management practices and consider tuning alert rules to reduce false positives for legitimate container operations

medium **investigate further** container_namespace_escape rule
 Rule may need refinement to distinguish between legitimate container operations and actual escape attempts

▲ 7 tool calls

```

get_alert_details → {"_id":"1fed393a-04c6-4335-9136-286c084de783","_index":"secureexec-alerts-180e3511-0586-42e2-9d80-a728cc5a3944","_primar... (8ms)
get_process_tree → {"nodes":[{"@timestamp":"2026-04-12T10:03:00.796399695+00:00","agent_id":"28980b93-fafc-447d-b37b-498a121cfbad","cmdline... (151ms)
get_endpoint_info → {"agent_id":"28980b93-fafc-447d-b37b-498a121cfbad","agent_version":"0.1.42","firewall_backend":"ebpf","hostname":"sitech... (7ms)
search_events → {"_shards":{"failed":0,"skipped":3,"successful":4,"total":4},"hits":{"hits":[{"_id":"sNs9gZ0Bia52l-T2v3d1","_index":".ds... (42ms)
get_alert_history → {"_shards":{"failed":0,"skipped":0,"successful":1,"total":1},"hits":{"hits":[{"_id":"1fed393a-04c6-4335-9136-286c084de78... (11ms)
search_events → {"_shards":{"failed":0,"skipped":3,"successful":4,"total":4},"hits":{"hits":[],"max_score":null,"total":{"relation":"eq"... (10ms)
search_events → {"_shards":{"failed":0,"skipped":3,"successful":4,"total":4},"hits":{"hits":[{"_id":"RNolgz0Bia52l-T28ZLI","_index":".ds... (36ms)
  
```

Рис. 2: AI-аналитик — расследование false-positive: AI объясняет, почему алерт по смене namespace — это легитимное поведение container runtime, и предлагает уточнение правила.

AI Case Summary True Positive 98% confidence
deepseek/deepseek-v3.2

Incident Summary

На хосте securehexec1 зафиксирована полномасштабная компрометация: злоумышленник получил доступ через SSH (сессия root@pts/1, установленная с IP 5.42.105.237) и запустил скрипт demo-attack.sh, который выполнил разведку сети через reverse shell-соединения на C2-сервер 198.51.100.1 по портам 4444, 22, 25 и 3389, а также запустил python3.12 для подключения к сервисным портам. Сразу после этого были созданы механизмы персистентности: cron-задача, systemd unit-файл, модификация SSH authorized_keys и создание sudoers-файла demo-backdoor для обеспечения постоянного привилегированного доступа. Все 12 алертов являются истинно-положительными и относятся к одной целенаправленной атаке с использованием скрипта demo-attack.sh, запущенного от root. Атака охватывает полный kill chain: начальный доступ (SSH), выполнение кода, разведку сети, закрепление (persistence) и, вероятно, эксфильтрацию данных.

Impact Assessment

Полная компрометация критического хоста securehexec1 с правами root. Злоумышленник имеет постоянный привилегированный доступ через SSH (authorized_keys), sudoers-бэкдор, systemd-сервис и cron-задачу. Возможна кража конфиденциальных данных, использование хоста для атак на другие системы в сети (lateral movement), а также использование в качестве спам-бота (проверка SMTP-порта 25). Хост необходимо немедленно изолировать.

Recommended Actions

urgent **isolate host** `securehexec1`
 Хост полностью скомпрометирован с правами root, активны reverse shell-соединения на C2-сервер 198.51.100.1. Необходимо немедленно изолировать хост для предотвращения дальнейшего ущерба и lateral movement.

urgent **kill tree** `2282f2c854e8dca63e7b4d652d91f445c8e4f5a281be17b983bcab595edc5b6b`
 Необходимо завершить все процессы, связанные с атакой: bash (pid 2512401), все дочерние timeout/bash reverse shell-процессы и python3.12, запущенные от demo-attack.sh.

high **block hash** `bc5945feb8bd26203ebfafea5ce1878bb2e32cb8fb50ab7ae395cfb1e1aaaf1`
 Заблокировать хеш исполняемого файла bash, использованного в атаке (если это не легитимный /usr/bin/bash, а подменённая версия).

urgent **investigate further** `198.51.100.1`
 IP-адрес C2-сервера, на который уходили reverse shell-соединения (порты 4444, 22, 25, 3389). Необходимо проверить в Threat Intelligence и заблокировать на фаерволе.

high **investigate further** `5.42.105.237`
 IP-адрес, с которого была установлена SSH-сессия root@pts/1. Необходимо проверить источник подключения и определить, был ли это легитимный доступ или атака на SSH.

urgent **review access** SSH ключи и sudoers на securehexec1
 Злоумышленник создал файл /etc/sudoers.d/demo-backdoor и модифицировал authorized_keys. Необходимо удалить все несанкционированные ключи и sudoers-конфигурации.

Рис. 3: AI case summary — связанные алерты с одного хоста в коротком окне группируются в один AI-кейс, поэтому вердикт, нарратив и рекомендации выдаются на инцидент, а не на каждый алерт по отдельности.

6. Реагирование

Детекция без реагирования — это генератор тикетов. В SecureExec реагирование живёт в той же консоли, что и детекция.

6.1 Сетевая изоляция хоста

Карантин скомпрометированного хоста — одним кликом. Два взаимозаменяемых backend (агент выбирает "auto", "ebpf" или "kmod"):

- **eBPF TC firewall** (по умолчанию). TC ingress/egress классификаторы с IHL-aware L4 offsets. Stateless TCP SYN filtering, IPv6 drop, loopback pass. Новые интерфейсы (контейнеры, VPN-туннели) автоматически подхватываются через netlink.
- **Kernel-модуль** (опционально, deb/rpm через DKMS). Те же whitelist-семантики, что и у eBPF-варианта. Пригодится на ядрах, где TC/eBPF ограничен.

Whitelist-правила живут в консоли: глобальные дефолты плюс per-endpoint оверрайды. Типовая политика: «разрешить SSH с jump host + сервер SecureExec + мониторинг; всё остальное — drop».

6.2 Блокировка процессов

- **Blocklist по hash** — добавили SHA-256, и exec такого бинаря по всему парку запрещается через fanotify.
- **Blocklist по path** — блокировка по шаблону пути (полезно против известных путей дропа малвари).
- Глобальный blocklist на организацию, с per-agent статусом синхронизации.

6.3 Kill process tree

Из алерта или из отдельной action-страницы — убить process_guid и всех его потомков. Агент ходит по таблице процессов BFS и шлёт сигналы; UI показывает, какие PID убиты.

6.4 Live response (read-only host-команды)

Агент предоставляет 14 read-only host-команд — чтение файлов, список процессов, journal, lsof/ss/lsmof, last и т.д. (полный список — в §5.3). Они доступны и людям-аналитикам в консоли, и AI-аналитику как типизированные тулзы — расследование может запрашивать живую host-side ground-truth (срез журнала, содержимое /etc/crontab, listening-сокеты) не выходя из кейса.

6.5 Уведомления и интеграции

Исходящие уведомления:

- **Slack** — rich mrkdwn-форматирование, severity-эмодзи, кликабельные ссылки на алерты.
- **Telegram** — HTML-форматирование, компактные структурные поля.
- **Email** — plain-text для шумных inbox.

Каждый канал фильтруется по правилу, severity и организации. Сервер также отдаёт webhooks — алерты разносятся в любую downstream SIEM / тикетинг.

7. UI расследований

The screenshot shows the SecureExec console interface. At the top, there's a navigation bar with 'Главная', 'О нас', 'Блог', 'Цены', a language dropdown set to 'Русский', and a 'Консоль' button. On the left, a sidebar lists various monitoring and alerting features. The main area is titled 'События' (Events) and features a search bar with the query 'e.g. event_type:process_create AND hostname:"server-1"'. Below the search bar are filters for event type (set to 'Все'), host (set to 'Все'), agent ID, and process GUID. A table displays a list of events with columns: ВРЕМЯ (Time), ТИП (Type), ХОСТ (Host), PID, ПРОЦЕСС (Process), and ДЕТАЛИ (Details). The events include actions like net_listen, process_exit, file_delete, process_fork, and process_create.

ВРЕМЯ	ТИП	ХОСТ	PID	ПРОЦЕСС	ДЕТАЛИ
04/21/2026, 08:10:10 PM	net_listen	secureexec1	2460588	secureexec-serv	::ffff:172.18.0.8:50051
04/21/2026, 08:10:09 PM	process_exit	secureexec1	2462592	postgres	
04/21/2026, 08:10:09 PM	file_delete	secureexec1	2459391	containerd-shim-runc-v2	/run/containerd/io.containerd.runtime.v2.task/moby/b88e145e2aee8a...
04/21/2026, 08:10:09 PM	file_delete	secureexec1	2947	dockerd	/var/run/docker/containerd/b88e145e2aee8a8af6a8229ef5b4f40deb...
04/21/2026, 08:10:09 PM	file_delete	secureexec1	2947	dockerd	/var/run/docker/containerd/b88e145e2aee8a8af6a8229ef5b4f40deb...
04/21/2026, 08:10:09 PM	process_exit	secureexec1	2462586	pg_isready	
04/21/2026, 08:10:09 PM	process_fork	secureexec1	2462592	postgres	postgres
04/21/2026, 08:10:09 PM	process_create	secureexec1	2462586	busybox	/bin/sh -c pg_isready -d \$(POSTGRES_DB) -U \$(POSTGRES_USER)

Рис. 4: Консоль — поток событий парка в реальном времени, поиск с синтаксисом Lucene, фильтры по полям, экспорт в JSON/CSV.

- **Dashboard** — счётчики в реальном времени, 7-дневные графики, топ типов событий, распределение по severity.

- **Endpoints** — список со статусом, версией агента, состоянием изоляции, состоянием kmod/eBPF firewall, последними алертами. Один клик — карантин.
- **Events** — full-text поиск (Elasticsearch) по всему парку с фильтрами по типу события, хосту, контейнеру, времени.
- **Process tree** — интерактивный граф «родитель–потомок», укоренённый в любом process_guid: вверх до init, вниз — по поддереву потомков.
- **Alerts** — детали алерта с привязанными исходными событиями, описанием правила и AI-отчётом, если он есть.
- **Cases** — объединение связанных алертов в один кейс с заметками и боковой панелью AI-чата; экспорт кейса (алерты + события + заметки + AI-отчёт) одним JSON для офлайн-разбора или хэндовера.
- **Rules** — встроенные и кастомные правила; suppression; blacklist; правила изоляции.
- **Organisation** — участники, RBAC, каналы уведомлений, конфигурация AI (провайдер, API-ключ, модель, режим live-response).

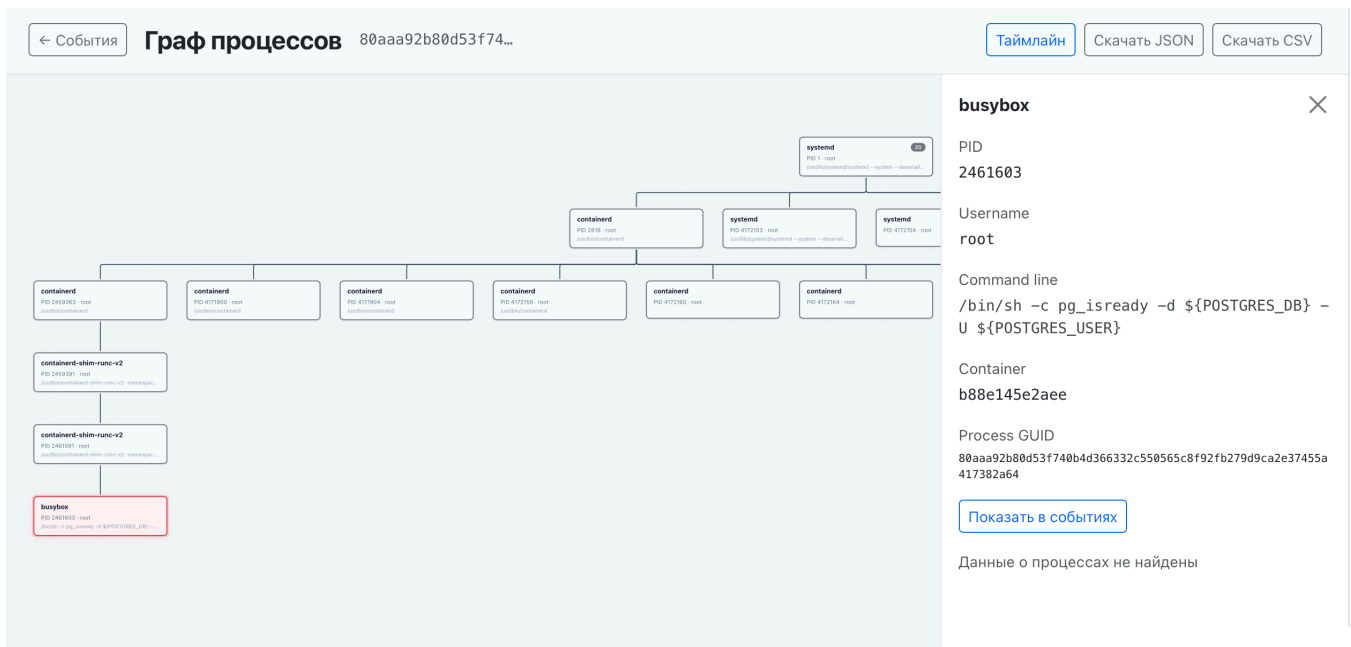


Рис. 5: Process tree — интерактивный граф родитель/потомок, укоренённый в любом process_guid: разворачивается вверх по цепочке до init или вниз по поддереву потомков.

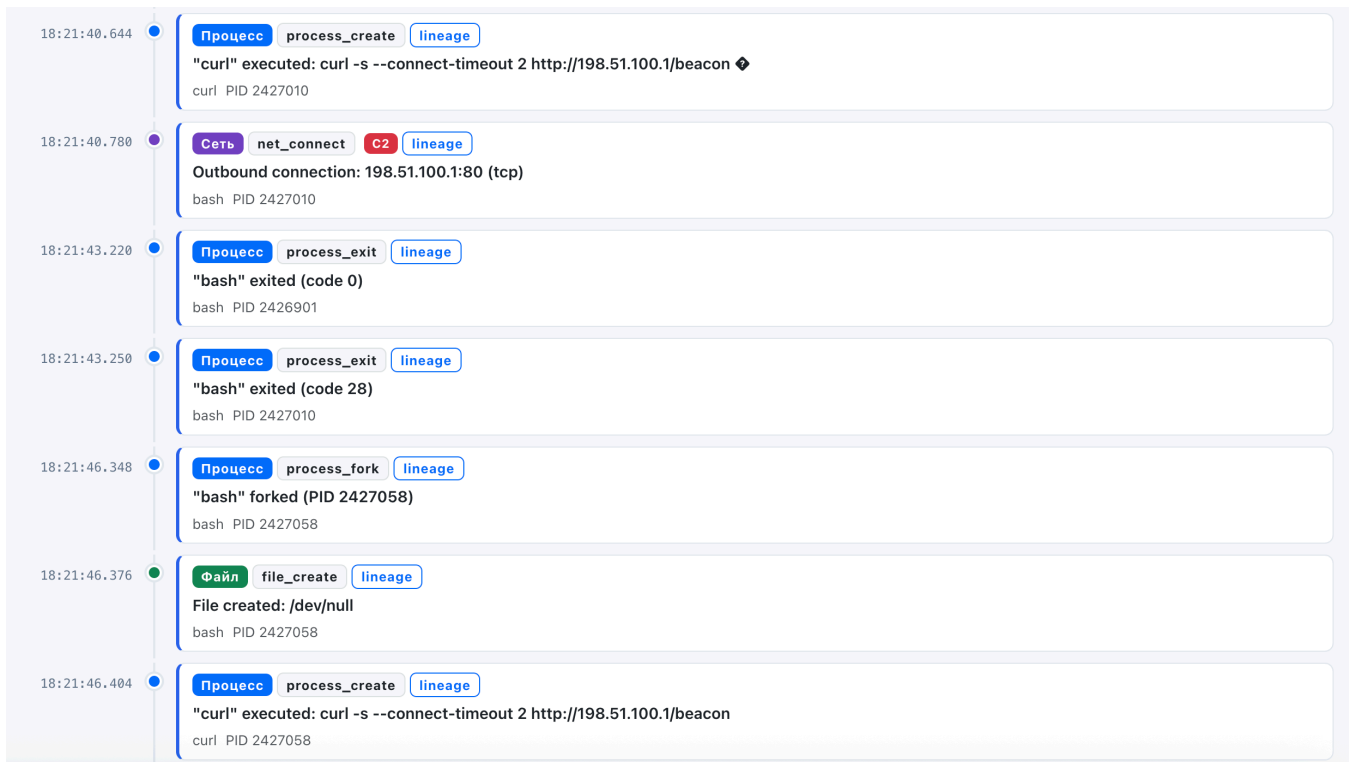


Рис. 6: Таймлайн событий вокруг алерта — то же представление, к которому ходит AI-аналитик, чтобы искать подтверждающие или оправдывающие данные: сетевые соединения, изменения файлов, смены привилегий.

8. Как установить

Два пути. Оба ставят вас в консоль с живой телеметрией заметно быстрее, чем за час.

8.1 Управляемое облако — 5 минут

1. Регистрация на securehex.ru.
2. В дашборде — копируете установочный one-liner.
3. Запускаете на хосте. Через секунды хост виден в консоли с heartbeat и живой телеметрией; AI-аналитик готов расследовать первый алерт.

Без CAPEX, без подъёма VM, без DevOps. Подписка на хост, ежемесячная оплата. Managed AI-аналитик включён — не нужно платить за OpenAI/Anthropic отдельно.

8.2 Self-hosted — control plane

Требования:

- **Хост control plane:** одна Linux VM (или небольшой Kubernetes-кластер — Helm chart в roadmap), docker и docker compose. Реалистичный сайзинг по размеру парка из расчёта ~10–20 evt/s sustained на хост и retention 7 дней:

Парк	vCPU	RAM	SSD
≤50	4	8 ГБ	200 ГБ
≤200	8	16 ГБ	1 ТБ
≤500	8	32 ГБ	2–4 ТБ
≤1000	16	64 ГБ	4–8 ТБ

Главный потребитель — Elasticsearch. Указанные размеры дают JVM heap (50% от RAM) достаточно места под bulk-индексацию без rejected bulks и оставляют запас под Postgres, gRPC ingest и веб-стек. Свыше ~1000 хостов на одном control plane — масштабируем Elasticsearch горизонтально или шардим по тенантам.

- **Агенты:** любое ядро Linux 5.10+ с BTF. Протестировано на Ubuntu 22.04/24.04, Amazon Linux 2023, Bottlerocket, Debian 12, RHEL/Rocky 9.

Всё упаковано в один `deploy/docker-compose.yml`:

```
nginx      - TLS termination (Let's Encrypt или ваш сертификат)
frontend  - Next.js веб-консоль
backend    - HTTP API (Rust)
server     - gRPC ingest (Rust) на :50051
postgres  - конфигурация, пользователи, правила, кейсы
elasticsearch - события, алерты
certbot    - опциональная автоматизация Let's Encrypt
```

Подъём — три шага:

1. Клонировать `deploy`-репозиторий, скопировать `.env.example` в `.env`, выставить `DOMAIN`, пароли, LLM endpoint.
2. `docker compose up -d`. На первом старте провижнутся Elasticsearch-шаблоны, прокатываются индексы, создаётся `admin`-пользователь.
3. Открыть `https://<ваш-домен>/` и выпустить первый сертификат агента из консоли.

Для production-нагрузок те же компоненты разворачиваются как Kubernetes Deployment-набор (Helm chart в roadmap на Q3 2026).

8.3 Self-hosted — агент (одна строка на любом Linux-хосте)

В консоли — копируете установочную команду для вашего дистрибутива. Она:

- Кладёт прекомпилированный статический musl-бинарь в `/opt/secureexec/bin/secureexec-agent`.
- Раскладывает конфиг и сертификаты в `/opt/secureexec/{etc,var}/`.
- Регистрирует и запускает `systemd`-сервис.

Агент читает `/opt/secureexec/var/secureexec-agent.json`:

```
{
  "backend_url": "https://your-server:50051",
  "tls_ca": "/opt/secureexec/etc/certs/ca.crt",
  "tls_client_cert": "/opt/secureexec/etc/certs/agent.crt",
  "tls_client_key": "/opt/secureexec/etc/certs/agent.key",
  "auth_token": "your-org-token"
}
```

Через секунды хост виден в консоли с `heartbeat`, первые события идут, правила начинают срабатывать.

8.4 Kubernetes

Агент спроектирован для запуска как **privileged DaemonSet** (`hostPID`, `hostNetwork`, `CAP_BPF` + `CAP_PERFMON` + `CAP_NET_ADMIN`). Готовый Helm chart на агент и control plane — в roadmap на Q3 2026; сейчас production-деплой используют DaemonSet-манифест, выведенный из `docker-compose`-конфигурации.

8.5 Удаление

Одна удалённая команда из консоли чисто удаляет агента: останавливает сервис, отвязывает eBPF-программы, убирает `ktod`, если был установлен, удаляет файлы. Перезагрузка не нужна.

9. Безопасность и границы доверия

- **mTLS везде** на north-south пути. У каждого агента свой сертификат; сервер не принимает никого другого.

- **Содержимое событий не покидает периметр клиента** в self-hosted-варианте, если оператор сам не указал внешний LLM endpoint. С локальной LLM (Bedrock, vLLM, Ollama) весь деплой остаётся внутри периметра, а аудит-trail whitepaper-уровня (правила, AI-промты, AI-вердикты, действия реагирования) целиком лежит на вашем железе.
- **Управляемое облако** живёт в защищённом облаке с изолированными tenant'ами, TLS 1.3 + AES-256 at rest и документированной политикой retention. Тот же AI-аудит-trail доступен вам в консоли.
- **Никакой телеметрии агента к вендору.** SecureExec не «звонит домой». Update-чек ходит в ваш control plane, а не в наш.
- **Least privilege в roadmap:** сегодня DaemonSet работает privileged для простоты; на Q3 2026 — минимальный набор capability (CAP_BPF, CAP_PERFMON, CAP_NET_ADMIN).
- **Секреты Kubernetes / env vars не собираются.** Process- телеметрия включает argv, но по умолчанию исключает envp — чтобы случайно не уехать на сервер с облачными credentials.

10. Что уже работает и что в roadmap

Доступно сегодня

- Linux eBPF-агент с полным набором событий из §4.1.
- gRPC/mTLS-транспорт, Elasticsearch для событий, PostgreSQL под конфигурацию и кейсы.
- **63 встроенные детекции** + кастомные правила + suppression.
- Реагирование: изоляция хоста (eBPF/kmod), блокировка процессов (fanotify), kill process tree, глобальный blocklist.
- **14 read-only live-response host-тулзов**, доступны и людям-аналитикам, и AI-агенту.
- Расследование: process tree, full-text поиск, детали алерта, экспорт кейса, **multi-turn AI-чат внутри кейса.**
- **AI-аналитик v1 (Phase 0)** — двухфазный gather/synthesize pipeline, структурные вердикты с полным аудит-trail, запускается on-demand из любого алерта или кейса.
- Уведомления — Slack, Telegram, Email; исходящие webhooks.
- Развёртывание control plane через Docker Compose; самообновление агента с SHA-256-верификацией; удаление агента из консоли.
- Console-i18n (English + Русский) и локаль-aware веб-UI.
- **Управляемое облако** с 5-минутным curl-онбордингом; включённый managed AI-аналитик.

Ближайший roadmap (Q3–Q4 2026)

- AI auto-triage (Phase 1) — каждый новый алерт выше порога расследуется автоматически; уверенные false positive суппрессируются.
- AI semi-autonomous response (Phase 2) — AI предлагает конкретные действия; auto-approve политики выполняют безопасные.
- Helm chart на control plane и агент; least-privilege DaemonSet.
- 200+ курируемых правил с AI-driven suppression для почти нулевого FP-rate.
- Глубже container-runtime: containerd/CRI-O, корреляция по OCI image provenance.
- Агенты Windows и macOS (уже есть в дереве; фокус поставки — Linux).

11. Коммерческое предложение

Подписочная модель, без перпетуальных лицензий, проходит как OPEX:

- **Управляемое облако.** Подписка на хост, помесечно. Managed AI-аналитик, поддержка, автообновления и бэкапы включены. Подходит стартапам, SMB и командам без выделенного DevOps.
- **Self-hosted.** Подписка на хост, годовая, разворачивается в вашей инфраструктуре (Docker Compose или Kubernetes). Air-gapped и закрытые сети поддерживаются. Подходит банкам, госсектору, телекомам и любым compliance-задачам, где данные не должны покидать периметр.

- **Агент** — open source под Apache 2.0 (github.com/secureexec/secureexec-agent). В eBPF-программах ядра стоит тег Dual MIT/GPL — это требование kernel-verifier для доступа к GPL-only BPF helpers, а не смена лицензии исходников.

Один и тот же продукт, те же SLA, тот же AI-аналитик в обеих редакциях — различается только то, где живёт control plane.

12. Как начать

Три простых пути — выбирайте любой:

- **5-минутный cloud-онбординг.** Регистрируетесь на secureexec.ru, запускаете один curl на тестовом хосте, смотрите, как AI-аналитик расследует первый алерт.
 - **30-минутное демо.** Расшариваем экран, проводим по готовому окружению с воспроизведением живой атаки, отвечаем на вопросы. Подходит, если хочется увидеть рассуждения AI-аналитика по реальному инциденту.
 - **Self-hosted ПOC.** Одна VM под control plane, один агент на тестовом хосте. Детектировать и реагировать начнёте в первый час.
-

13. Контакты

- Андрей Сметанин — Founder · andrey@secureexec.ru · <https://t.me/irqlevel>
- Сайт · <https://secureexec.ru>
- Продажи · sales@secureexec.ru
- Поддержка · support@secureexec.ru
- Исходники агента (Apache 2.0) · <https://github.com/secureexec/secureexec-agent>