Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Systemd на максимум: таймеры и sandboxing без Docker

Как мы переводим регламентные задачи с cron на таймеры и изолируем сервисы средствами PID 1

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Регламентные задачи клиента — бэкапы БД, обмены 1С, очистки — живут в crontab без логов, ретраев и зависимостей, а прикладные сервисы работают от root с полным доступом к ФС. Упавший бэкап обнаруживается в момент восстановления, компрометация одного сервиса отдаёт весь сервер. Мы переводим задачи на systemd-таймеры и зажимаем сервисы штатным sandboxing — без Docker.

Почему это важно бизнесу

- Потерянный ночной бэкап без алерта обнаруживается при аварии, когда восстанавливать уже нечем, — простой от суток
- Сервис от root без изоляции: одна уязвимость в приложении — и атакующий получает весь сервер с данными
- Одновременный старт задач на парке серверов кладёт диски и канал — прод тормозит в самый неподходящий момент
- Разбор инцидента без централизованных логов занимает часы вместо минут — счёт за простой растёт
- Всё штатными средствами ОС: без лицензий, без контейнерной инфраструктуры и её сопровождения

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

v252+

минимальная версия systemd на парке (Debian 12 / RHEL 9): весь набор директив уже доступен

systemd, Debian 12/RHEL 9

30+

директив изоляции в systemd.exec: ProtectSystem, PrivateTmp, SystemCallFilter и другие

по докам systemd.exec(5)

≤3.0

целевой exposure-балл systemd-analyze security для каждого прикладного сервиса (шкала 0–10)

наш стандарт

600 с

RandomizedDelaySec= в каждом таймере — размазываем старты по парку, гасим thundering herd

по докам systemd.timer(5)

1 ГБ

SystemMaxUse= плюс MaxRetentionSec=30day в journald.conf — журнал не съедает диск

по докам journald.conf(5)

2 нед

параллельный прогон таймеров рядом с cron до отключения старых crontab-строк

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Миграция регламентных задач: cron → systemd-таймеры

Что настраиваем

Linux-серверы клиента (Debian 12 / Ubuntu 24.04, systemd 252–255): бэкапы PostgreSQL, обмены 1С, очистки и выгрузки

Как мы это делаем

1. Инвентаризация crontab всех пользователей и /etc/cron.d; каждой строке — пара имя.service (Type=oneshot, User=, IOSchedulingClass=idle) + имя.timer
2. В таймере: OnCalendar= вместо cron-маски (проверяем systemd-analyze calendar), Persistent=true для пропущенных запусков, RandomizedDelaySec=600
3. Зависимости: After=postgresql.service в бэкап-юните; на ошибку — OnFailure=alert@%n.service с уведомлением дежурному в Telegram
4. Тяжёлым задачам — лимиты CPUQuota=50% и MemoryMax=512M, чтобы бэкап не отбирал ресурсы у прода
5. Параллельный прогон с cron две недели, сверка journalctl -u по каждому юниту, затем отключение cron-строк

РЕЗУЛЬТАТ

Все задачи видны одной командой systemctl list-timers, каждый запуск — с полным логом в journald и алертом при падении. Пропущенные из-за ребута запуски выполняются при загрузке (Persistent=true), пики одновременного старта по парку размазаны RandomizedDelaySec.

КЛЮЧЕВОЙ НЮАНС

OnCalendar= считается в таймзоне сервера и не равен cron-маске один в один: перед вводом прогоняем каждое выражение через systemd-analyze calendar --iterations=5 и смотрим даты следующих запусков.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Sandboxing прикладных сервисов штатными директивами

Что настраиваем

Самописные приложения и агенты обмена на серверах клиентов: веб-бэкенды, интеграции 1С — всё, что исторически писалось под root

Как мы это делаем

1. Базовый drop-in через systemctl edit: отдельный User=/Group=, NoNewPrivileges=true, ProtectSystem=strict, ProtectHome=true, PrivateTmp=true
2. Запись — только явно: ReadWritePaths= или StateDirectory=/LogsDirectory= (каталоги создаёт сам systemd); для stateless-агентов — DynamicUser=yes
3. Syscall и привилегии: SystemCallFilter=@system-service, SystemCallArchitectures=native, CapabilityBoundingSet=, RestrictAddressFamilies=AF_INET AF_INET6 AF_UNIX
4. Ядро: ProtectKernelTunables/Modules=true, ProtectControlGroups=true, MemoryDenyWriteExecute=true (кроме JIT), LockPersonality, RestrictRealtime, RestrictSUIDSGID
5. Верификация: systemd-analyze security до/после (типично 9.6 UNSAFE → 1.8–3.0), затем регресс-тест приложения под нагрузкой

РЕЗУЛЬТАТ

Эксплойт в приложении упирается в read-only ФС, отфильтрованные syscalls и пустой набор capabilities — горизонтальное перемещение по серверу блокируется на уровне PID 1. Изоляция уровня контейнера без Docker-демона, лишнего рантайма и переупаковки приложения.

КЛЮЧЕВОЙ НЮАНС

Директивы включаем по одной и смотрим journald: SystemCallFilter убивает процесс SIGSYS без внятной ошибки, а MemoryDenyWriteExecute несовместим с JIT (Java, Node.js) — такие сервисы жмём остальным набором.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Журнал и контроль: journald, алерты, экспорт в Loki

Что настраиваем

Все таймеры и сервисы парка: персистентный журнал на хосте, алертинг об отказах, централизованное хранение вне сервера

Как мы это делаем

1. journald.conf: Storage=persistent, SystemMaxUse=1G, MaxRetentionSec=30day — журнал переживает ребут и не съедает диск
2. Разбор инцидентов: journalctl -u юнит -p err --since, вывод -o json в jq; журналы прошлых загрузок — ключом -b -1
3. Экспорт в Grafana Loki: Alloy/Promtail читает journal, relabel __journal__systemd_unit → метка unit, хранение 90 дней вне хоста
4. Мониторинг: systemctl --failed и list-timers в Zabbix; шаблон alert@.service через OnFailure= шлёт имя упавшего юнита в Telegram

РЕЗУЛЬТАТ

Инцидент разбирается по индексированному журналу за минуты, история хранится вне хоста и не теряется при отказе диска. Дежурный узнаёт об упавшей задаче из алерта в течение минуты, а не от клиента после сбоя.

КЛЮЧЕВОЙ НЮАНС

Без Storage=persistent журнал живёт в tmpfs /run/log/journal и исчезает при ребуте — лишь с v259 персистентность стала умолчанием, поэтому на парке v252–255 каталог /var/log/journal создаём сами.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **ProtectSystem=strict без ReadWritePaths**
  ФС становится read-only, сервис падает на первой записи; заранее находим пути записи по логам и открываем их точечно через ReadWritePaths=.

- ✕ **OnCalendar ≠ cron-маска**
  Синтаксис календаря другой, время — в таймзоне сервера; каждое выражение проверяем systemd-analyze calendar до ввода в бой.

- ✕ **Restart= в Type=oneshot до v244**
  На systemd старше v244 юнит с таким сочетанием не загрузится; Restart=always/on-success запрещены с oneshot и в новых версиях — ретраи строим через O…

- ✕ **PrivateTmp рвёт обмен через /tmp**
  У сервиса свой изолированный /tmp: файлы от других процессов «исчезают»; обмен переносим в /var/lib с явным ReadWritePaths=.

- ✕ **DynamicUser и файлы вне StateDirectory**
  UID выделяется динамически на каждый запуск: файлы, созданные вне StateDirectory/CacheDirectory, после рестарта теряют владельца.

- ✕ **MemoryDenyWriteExecute против JIT**
  Java, Node.js, .NET требуют W+X-память и падают; для них директиву не включаем, компенсируя SystemCallFilter и NoNewPrivileges.

- ✕ **journald без лимитов размера**
  Persistent-журнал разрастается до заполнения /var; ставим SystemMaxUse=1G, MaxRetentionSec=30day и контролируем journalctl --disk-usage.

- ✕ **Таймеры без RandomizedDelaySec**
  Один OnCalendar= на всём парке — одновременный старт бэкапов кладёт СХД и сеть; закладываем RandomizedDelaySec=600 в шаблон таймера.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Критичные задачи (бэкапы, обмены) — на таймеры с Persistent=true
- Во всех прикладных юнитах: свой User=, NoNewPrivileges=true, PrivateTmp=true
- journald: Storage=persistent + SystemMaxUse=1G
- systemctl --failed и list-timers — в ежедневную проверку

НОРМАЛЬНО

- RandomizedDelaySec=600 и зависимости After=/Requires= на всех таймерах
- ProtectSystem=strict + ReadWritePaths/StateDirectory для сервисов
- OnFailure=alert@%n — алерт в Telegram/почту по каждому падению
- Лимиты CPUQuota= и MemoryMax= на тяжёлых регламентных задачах

ХОРОШО

- Exposure ≤3.0 по systemd-analyze security у всех прикладных сервисов
- SystemCallFilter=@system-service и пустой CapabilityBoundingSet=
- Журналы — в Loki/ELK, retention 90 дней вне хоста
- Stateless-агенты — на DynamicUser=yes и portable services

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Все регламентные задачи видны в systemctl list-timers, а не разбросаны по crontab пользователей?
- [ ] Выполнится ли пропущенный из-за ребута бэкап автоматически (Persistent=true на критичных таймерах)?
- [ ] Узнаёт ли дежурный об упавшей задаче в течение часа — или это выяснится при восстановлении?
- [ ] Каждый прикладной сервис работает от своего User=, а не от root?
- [ ] Прогнан ли systemd-analyze security по сервисам и известен ли exposure-балл каждого?
- [ ] Журнал персистентный (Storage=persistent) и ограничен по размеру (SystemMaxUse=)?
- [ ] Стоят ли MemoryMax=/CPUQuota= на тяжёлых задачах, чтобы бэкап не тормозил прод?
- [ ] Разнесены ли старты задач по парку через RandomizedDelaySec, нет ли пиков одновременного запуска?
- [ ] Уходят ли логи во внешнее хранилище (Loki/ELK) и переживут ли они отказ диска сервера?

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 crontab и юнитов Linux-парка, миграция регламентных задач на таймеры с алертингом — под ключ
- Hardening сервисов до exposure ≤3.0: drop-in шаблоны, syscall-фильтры, регресс-тест после включения
- Централизация логов: journald → Loki/Grafana, дашборды и алерты по failed-юнитам
- Поддержка Linux-серверов с мониторингом таймеров и сервисов, реакция по SLA

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 systemd.timer(5) — OnCalendar, Persistent, RandomizedDelaySec (freedesktop.org — v259)

02 systemd.exec(5) — Sandboxing directives (freedesktop.org — v259)

03 systemd-analyze(1) — security, calendar (freedesktop.org — v259)

04 journald.conf(5) — Storage, SystemMaxUse, MaxRetentionSec (freedesktop.org — v259)

05 systemd.resource-control(5) — CPUQuota, MemoryMax (freedesktop.org — v259)

06 systemd.service(5) — Type=oneshot, Restart=, OnFailure= (freedesktop.org — v259)

07 Наш drop-in шаблон hardening.conf для прикладных юнитов (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.