ODYSSEY LINUX · СПРАВОЧНАЯ ДОКУМЕНТАЦИЯ

Odyssey Control Center — Практическое руководство

Каждая панель, раздел за разделом. Что она делает, что меняет, и куда нажимать — написано для тех, кто никогда её не использовал.

Launch Edition

Don’t trust me. Verify me.

— nobody

Odyssey Linux — Odyssey Control Center: Практическое руководство
Launch Edition. Описывает Control Center в том виде, в каком он поставляется на ISO-образе Launch Edition — десять полных панелей, от Boot до About, плюс временная глава для модуля драйвера NVIDIA (видна только на оборудовании NVIDIA, ожидает скриншотов с машины, у которой оно есть).

Copyright © 2026 the Odyssey Linux project.
Опубликовано под лицензией Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).

Это руководство — практическое дополнение к Security and Hardening Guide, которое подробно рассматривает панели Firewall и AppArmor — здесь вы найдёте пошаговое визуальное руководство и ссылку на остальное.

Каждый экран, кнопка и поведение, описанные здесь, взяты из реального исходного кода Control Center и реальных скриншотов, а не по памяти.

Подписи кнопок, названия панелей и вкладок, пути к файлам и команды остаются на английском, потому что это реальный интерфейс программы: искать их переведёнными в Control Center бесполезно — вы их не найдёте.

СОДЕРЖАНИЕ

1 Что такое Control Center 1
1.1 Паттерны, которые вы увидите в каждой панели 2
2 Boot 3
2.1 Визуальное руководство — найти другую операционную систему и задать систему по умолчанию 3
2.2 Визуальное руководство — выбор того, что загружается по умолчанию 5
2.3 Визуальное руководство — вкладка EFI 8
2.4 Параметры ядра 9
2.5 Advanced 9
3 Kernel 11
3.1 Визуальное руководство — предустановки 11
3.2 Визуальное руководство — SYSCTL, группа за группой 12
3.3 Визуальное руководство — CPU governor 14
3.4 Визуальное руководство — hugepages 15
3.5 Визуальное руководство — modules 16
3.6 Advanced 17
4 Memory & Swap 19
4.1 Визуальное руководство 19
4.2 Advanced 20
5 Firewall 22
5.1 Визуальное руководство 22
5.2 Advanced 23
6 AppArmor 24
6.1 Визуальное руководство — STATUS 24
6.2 Визуальное руководство — PROFILES 25
6.3 Визуальное руководство — COMMUNITY 26
6.4 Визуальное руководство — VIOLATIONS 28
6.5 Визуальное руководство — EDITOR 29
6.6 Advanced 30
7 Privacy 32
7.1 Визуальное руководство — SearXNG, шаг за шагом 32
7.2 Визуальное руководство — Tor, шаг за шагом 35
7.3 Визуальное руководство — объединяя их 37
7.4 Advanced 38
8 Containers 40
8.1 Визуальное руководство — движок 40
8.2 Визуальное руководство — rootless vs. rootful, и пустое состояние 42
8.3 Визуальное руководство — отобранный каталог, от начала до конца 43
8.4 Визуальное руководство — список контейнеров и его действия 46
8.5 Визуальное руководство — запуск контейнера и его открытие 47
8.6 Использование диска 48
8.7 Advanced 48
9 Gaming Mode 50
9.1 Визуальное руководство — главный переключатель 50
9.2 Визуальное руководство — Packages и Steam 51
9.3 GameMode и MangoHud 53
9.4 Proton 54
9.5 Advanced 54
10 Telemetry 56
10.1 Визуальное руководство 56
10.2 Advanced 58

11 About 60
11.1 Визуальное руководство 60
11.2 Copy report 61
11.3 Advanced 61
12 NVIDIA driver 63

CHAPTER 1

Что такое Control Center

Odyssey Control Center — это десять небольших независимых программ за одним лаунчером — по одной на каждую область системы: Boot, Kernel, Memory & Swap, Firewall, AppArmor, Privacy, Containers, Gaming Mode, Telemetry, About, и (на оборудовании NVIDIA) модуль драйвера NVIDIA.

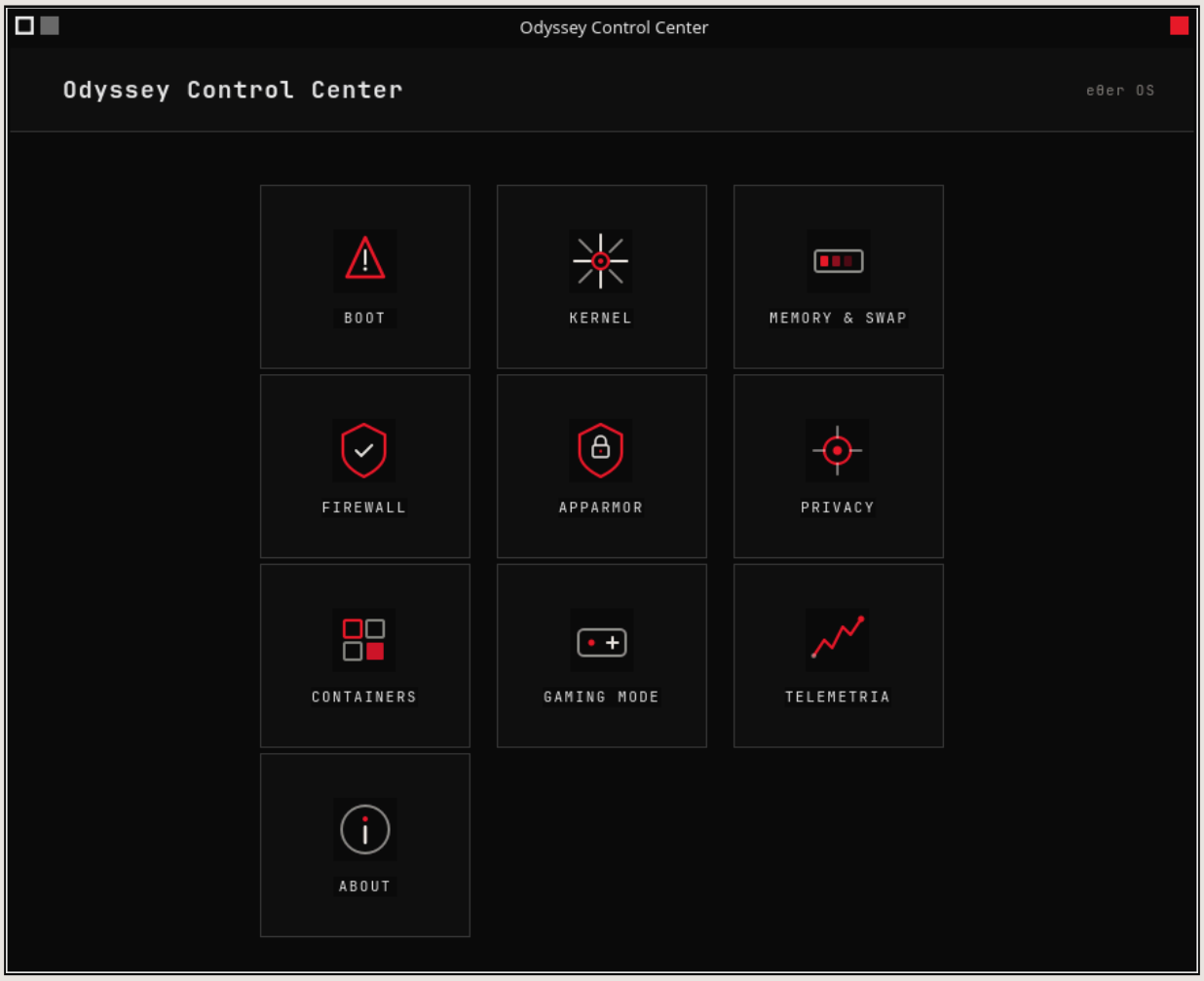

FIG · Главный экран Control Center — по одной плитке на каждую панель.

Каждая панель — это тонкий графический слой поверх настоящего механизма Linux под ним: конфигурации GRUB, собственного интерфейса sysctl ядра, `nftables`, `podman`. Панель никогда не выдумывает собственное скрытое состояние — она читает реальные файлы и команды, показывает их вам и записывает именно то, о чём вы попросили. Ничего из того, что она делает, не скрыто, и всё, что она делает, можно сделать и вручную, из терминала, с помощью тех же самых команд, которые это руководство показывает рядом с каждой кнопкой.

Это практическое руководство, а не справочник по каждой опции — оно отвечает на вопрос «на что нажать и что произойдёт, если я это сделаю». Каждая глава построена одинаково, в два этапа: сначала **визуальное руководство**, скриншот за скриншотом, кнопка за кнопкой, написанное для тех, кто никогда не открывал Control Center — затем раздел **ADVANCED**, отмеченный чёрным

баннером, который переходит к деталям: точные пути к файлам, реальные команды, которые выполняет каждая кнопка под капотом, и что делать из терминала, если вы хотите полностью пропустить панель. Читайте только визуальное руководство, или сразу переходите к Advanced, если вы уже знакомы с Linux — обе части самодостаточны.

Для двух панелей с реальными последствиями для безопасности — Firewall и AppArmor — подробный справочник — это **Security and Hardening Guide**; здесь вы найдёте визуальное руководство, самое важное и ссылку на остальное.

1.1 Паттерны, которые вы увидите в каждой панели

Несколько соглашений повторяются во всех десяти панелях — выучить их один раз означает выучить для всех.

ЗАПРОС ПАРОЛЯ — ТОЛЬКО ОДИН РАЗ НА ПАНЕЛЬ

Большинству панелей нужно читать или записывать файлы, принадлежащие root — конфигурацию GRUB, sysctl, набор правил файрвола. В первый раз, когда панели это нужно, вы получаете одно диалоговое окно аутентификации. Панели построены так, чтобы группировать свои операции чтения: открытие Boot, например, читает конфигурацию GRUB **и** состояние прошивки EFI за один запрос, поэтому вы вводите пароль один раз, а не дважды. Повторное открытие панели в той же сессии не запрашивает пароль снова; последующие действия (применение изменения, перезагрузка) показывают свой собственный прогресс в журнале самой панели.

ИНДИКАТОР ШАГОВ

Панели, которые проводят вас через установку чего-либо, прежде чем это станет пригодным к использованию — os-prober в Boot, SearXNG и Tor в Privacy, движок контейнеров в Containers — показывают вверху небольшой пронумерованный индикатор: ряд шагов, текущий выделен, и одна кнопка действия, которая всегда выполняет **следующий шаг**, каким бы он ни был. Вам никогда не придётся гадать, на что нажать первым.

ЖУРНАЛ ВНИЗУ

Каждая панель заканчивается небольшим прокручиваемым журналом. Это не украшение — это точная команда, которая была выполнена, её вывод и цветной результат (зелёный для успеха, красный для сбоя). Если панель делает что-то, чего вы не понимаете, именно в журнале находится настоящий ответ.

TRY VS. MAKE PERSISTENT

Повторяющаяся пара кнопок, особенно в Kernel и Memory: **TRY NOW** (или **APPLY NOW**) немедленно меняет работающую систему и откатывается при следующей перезагрузке, если вы больше ничего не сделаете. **MAKE PERSISTENT** записывает то же значение в конфигурационный файл под `/etc/`, чтобы оно пережило и перезагрузки. Это разделение существует намеренно — попробовать значение ничего не стоит, и ничто не постоянно, пока вы не скажете это дважды.

CHAPTER 2

Boot

У панели Boot есть две вкладки на любой машине, которая загружается в режиме UEFI: **GRUB** (само меню загрузки) и **EFI** (уровень прошивки под ним). На более старых машинах, только с BIOS, существует лишь GRUB, потому что уровня EFI под ним просто нет.

> ДВА УРОВНЯ, А НЕ ОДИН
>
> **GRUB** — это меню загрузки, которое вы видите на экране — оно решает, какая операционная система запускается, когда сам GRUB уже работает. **Прошивка EFI/UEFI** находится на один уровень ниже GRUB — она решает, какому загрузчику машина вообще передаёт управление в первую очередь, ещё до того, как загрузился сам GRUB. Большинство людей никогда не трогают вкладку EFI; она существует для случаев, которые GRUB не может охватить, например для второй установки Windows, которая независимо управляет своей собственной записью загрузки.

2.1 Визуальное руководство — найти другую операционную систему и задать систему по умолчанию

Вкладка GRUB открывается в разделе **Other OS detection**. Odyssey не сканирует другие системы по умолчанию — в большинстве установок на диске есть только Odyssey, а сканирование напрасно тратит время при каждой регенерации меню загрузки. Включить это — управляемая последовательность из трёх шагов, показанная как небольшой пронумерованный индикатор.

1. **`Installed`**

   Инструмент сканирования, os-prober, ещё не установлен в системе. Одна кнопка: **INSTALL OS-PROBER**.

2. **`Enable scan`**

   Инструмент установлен, но GRUB всё ещё настроен игнорировать его — это значение по умолчанию в Odyssey. Одна кнопка: **ENABLE SCAN**.

3. **`Scan`**

   Теперь GRUB будет его использовать. Одна кнопка, которую вы будете использовать снова каждый раз, когда добавляете или удаляете другую систему в будущем: **SCAN & REGENERATE**.

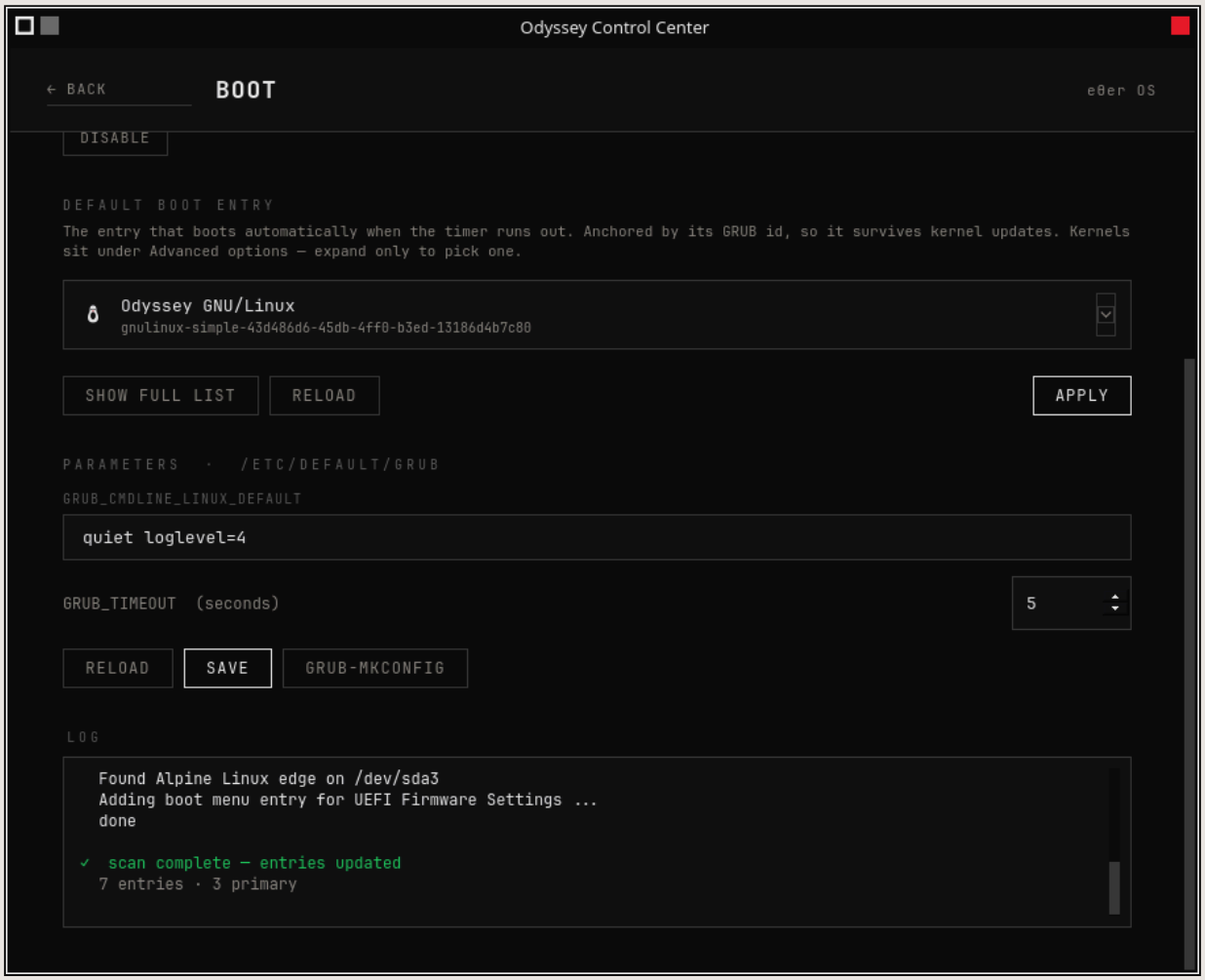

FIG · Все три шага зелёные — os-prober установлен, сканирование включено, готово. Единственная кнопка действия всегда соответствует текущему шагу.

Пройдите по шагам (каждый снова запрашивает аутентификацию только при необходимости) и запустите сканирование. Панель показывает вам точно, что она нашла, прежде чем вы покинете раздел:

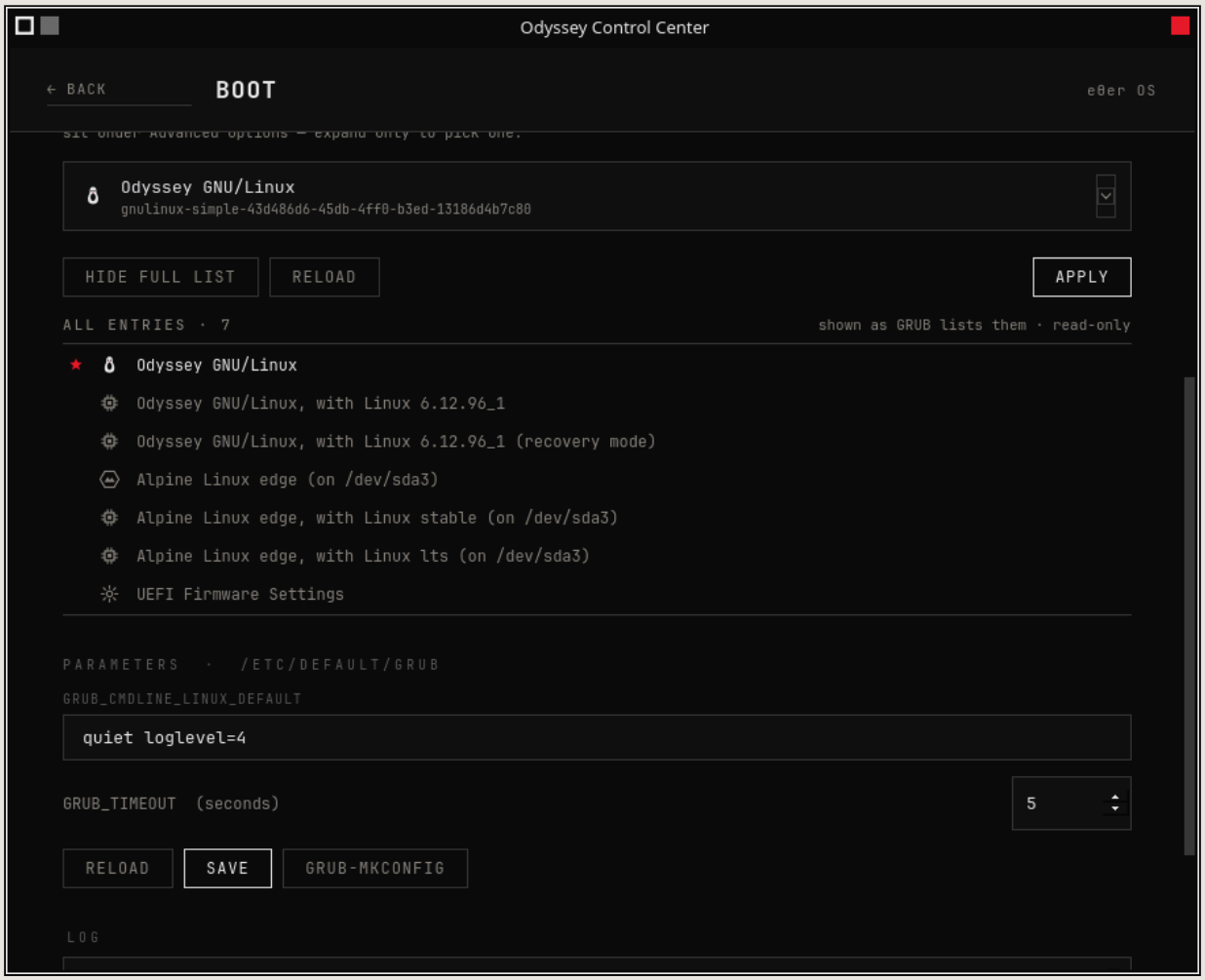

FIG · Сканирование завершено: Alpine Linux найден на /dev/sda3, вместе с собственными записями Odyssey и записью UEFI Firmware Settings — 7 записей, 3 основные.

Журнал подтверждает это простыми словами — **«scan complete — entries updated · 7 entries · 3 primary»**. Всё, что теперь находит os-prober, автоматически появляется в разделе ниже, готовое к выбору.

2.2 Визуальное руководство — выбор того, что загружается по умолчанию

Прокрутите до **Default boot entry**. Там показана текущая запись по умолчанию в виде строки; клик по ней открывает выпадающий список со всем, что можно выбрать вместо неё.

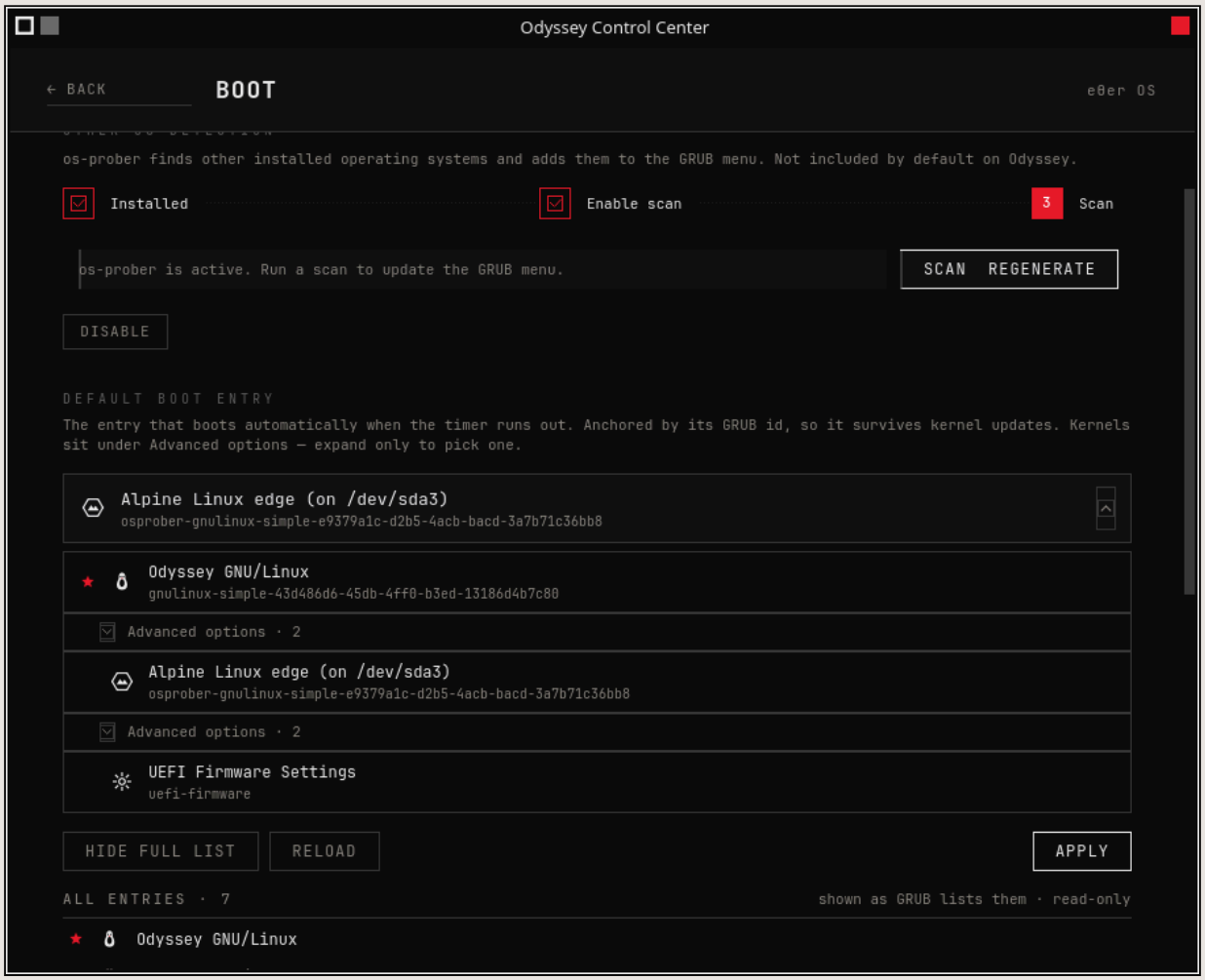

FIG · Открытый выпадающий список с выбранным Alpine Linux как ожидающий выбор — ещё не применённый.

Выбор записи из списка **ставит выбор в ожидание** — он ещё не действует. Видно, что он только в ожидании, потому что поле по-прежнему показывает новый выбор, но ничего не было записано. Кнопка, которая подтверждает его — **APPLY**, справа:

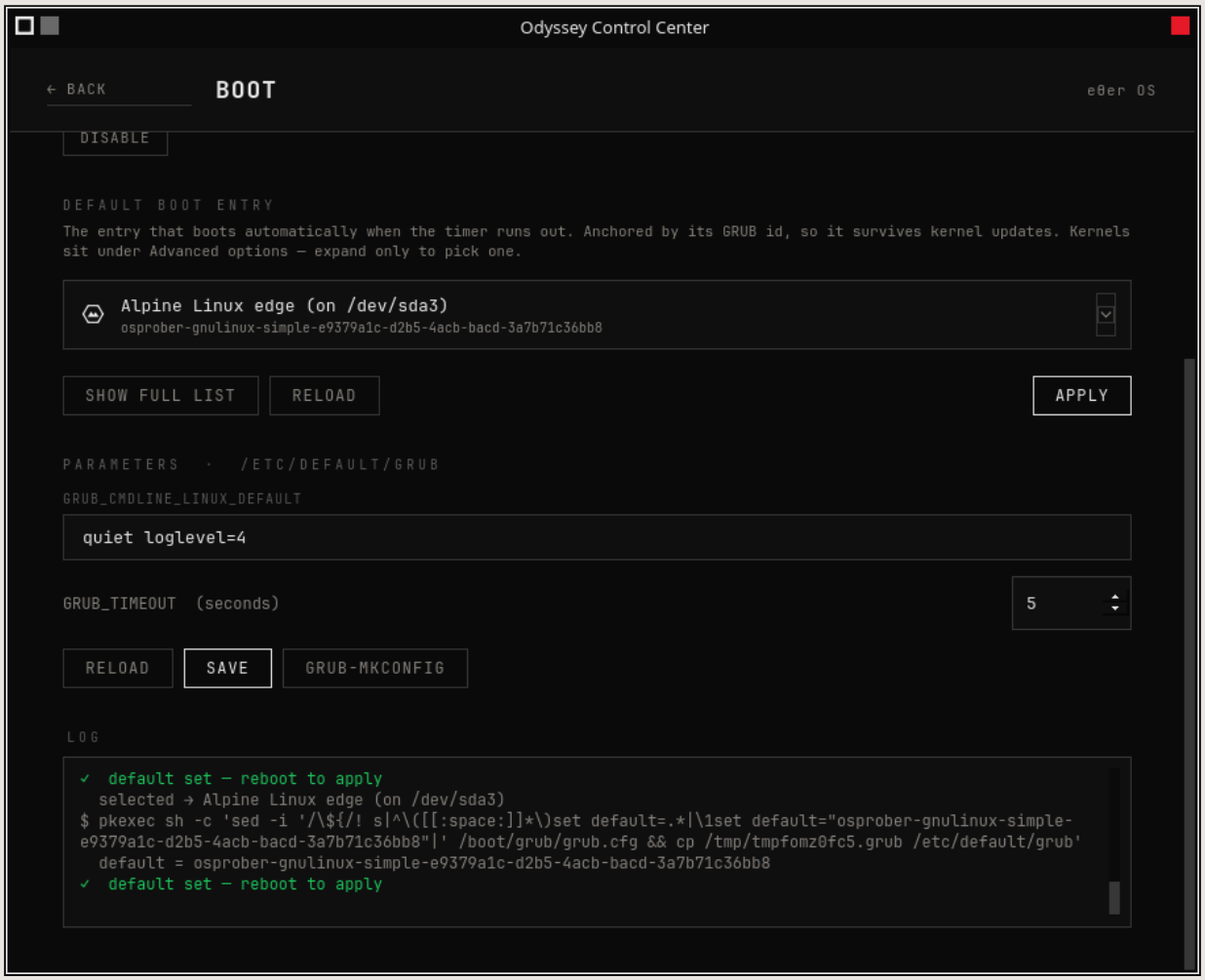

FIG · После APPLY: журнал подтверждает «default set — reboot to apply», а переключатель теперь показывает Alpine Linux как активную систему по умолчанию.

WHAT	Boot → вкладка GRUB → нажмите на текущую запись по умолчанию → выберите другую из выпадающего списка → нажмите **APPLY**.
WHY	У вас двойная загрузка, и вы хотите, чтобы автоматически запускалась другая система, или обновление ядра оставило выбранным старое ядро, а вы хотите вернуться к последнему.
RESULT	Выбранная запись автоматически загружается начиная со следующей перезагрузки. Изменение привязано к стабильному id этой записи, поэтому оно переживёт будущие обновления ядра.
RISK	Никакого для ваших данных — в худшем случае вы выберете не ту запись и вернётесь, чтобы снова её изменить.

SHOW FULL LIST показывает каждую запись точно так, как их перечисляет сам GRUB — только для чтения, сгруппированные так, что варианты для отдельных ядер находятся под раскрывающейся строкой «Advanced options», а не загромождают основной список. **RELOAD** перечитывает конфигурацию GRUB, если вы думаете, что она изменилась вне панели.

2.3 Визуальное руководство — вкладка EFI

Нажмите на вкладку **EFI** вверху. Это уровень прошивки — то, что показывает меню загрузки самой вашей материнской платы ещё до того, как вообще запустилась какая-либо операционная система.

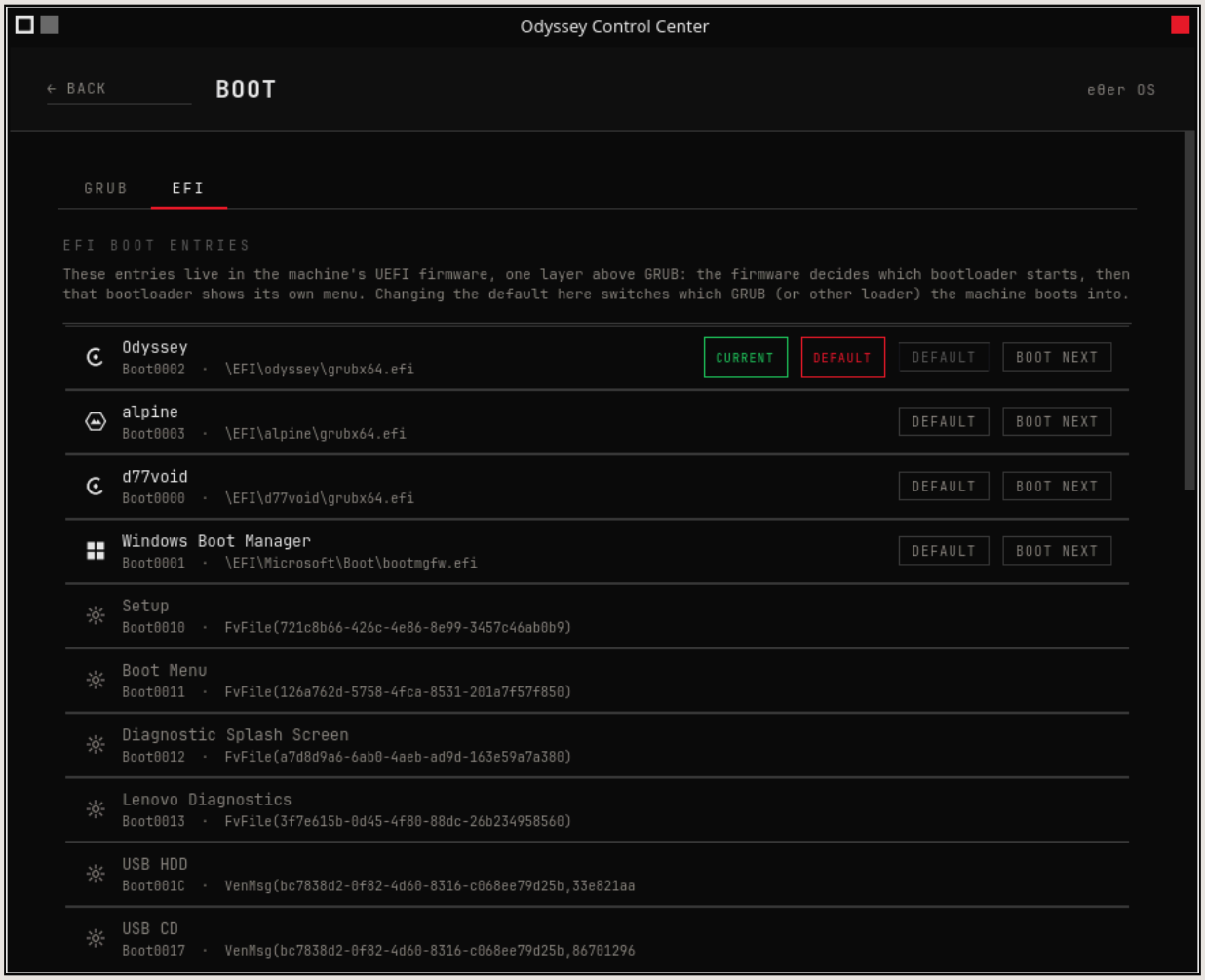

FIG · Вкладка EFI: Odyssey отмечен как CURRENT и DEFAULT зелёным/красным, alpine, d77void и Windows Boot Manager, каждый со своими кнопками DEFAULT / BOOT NEXT, плюс служебные записи прошивки ниже.

Каждая строка — это запись прошивки — её имя, её id GRUB (`Boot0002` и так далее), и точный файл `.efi`, на который она указывает. Две кнопки на строку:

КНОПКА	ЧТО ОНА ДЕЛАЕТ
DEFAULT	Делает эту запись постоянной записью прошивки по умолчанию — она загружается при каждой перезагрузке начиная с этого момента.
BOOT NEXT	Разовое исключение — эта запись загружается только при **следующей** перезагрузке, после чего обычный порядок автоматически возобновляется. Пока исключение ожидает выполнения, вверху появляется баннер с кнопкой CANCEL.

WHAT	Boot → вкладка EFI → найдите нужную запись (например, «Odyssey» или «Windows Boot Manager») → нажмите DEFAULT, или BOOT NEXT для разового исключения.
WHY	Если само меню загрузки вашей материнской платы полностью запускает не тот загрузчик — это не настройка GRUB, а настройка прошивки — это единственное место, которое это исправляет.
RESULT	DEFAULT: запись перемещается в начало порядка загрузки прошивки на постоянной основе. BOOT NEXT: она загружается один раз, затем откатывается.
RISK	Низкий, и всегда восстановимый с этой же вкладки или с самого экрана настройки UEFI вашей материнской платы (обычно Del или F2 при включении), если что-то загружается неожиданно.

2.4 Параметры ядра

Последний раздел вкладки GRUB — это командная строка ядра в чистом виде — редактируется как обычное текстовое поле, без необходимости синтаксиса GRUB — и таймаут меню в секундах. **SAVE** записывает их в `/etc/default/grub`; **GRUB-MKCONFIG** заново создаёт настоящее меню загрузки из этого файла — шаг, который делает сохранённый параметр действующим при следующей загрузке. Панель всегда повторно применяет ваш текущий выбор по умолчанию при регенерации, поэтому это никогда не сбрасывает, какая система загружается по умолчанию.

```
▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.
```

2.5 Advanced

ЧТО ПРОИСХОДИТ НА САМОМ ДЕЛЕ

Конфигурация GRUB в Odyssey — **плоская** — без блоков `submenu`. Каждое ядро — это `menuentry` верхнего уровня, что означает, что стабильный id (а не числовая позиция) всегда может корректно его выбрать, даже между обновлениями ядра. Группировка «Advanced options», которую вы видите в переключателе — это чистое визуальное удобство, построенное путём сопоставления общего UUID между простой записью операционной системы и её вариантами для разных ядер — сам файл GRUB не имеет такой группировки.

Установка значения по умолчанию записывает в двух местах в рамках одного действия: постоянный `GRUB_DEFAULT` в `/etc/default/grub`, и — поскольку `grub.cfg` может быть не сразу перегенерирован — прямое и точечное изменение через `sed` активной строки `set default="..."` в самом `/boot/grub/grub.cfg`, минуя одноразовую ветку `${next_entry}`, которую использует `grub-reboot`. Именно поэтому значение по умолчанию вступает в силу при следующей загрузке без предварительного принудительного полного `grub-mkconfig`.

```
$ cat /etc/default/grub | grep GRUB_DEFAULT
$ grep 'set default=' /boot/grub/grub.cfg
```

OS-PROBER ВРУЧНУЮ

```
$ sudo xbps-install -y os-prober
# включает сканирование — изменяет GRUB_DISABLE_OS_PROBER в /etc/default/grub
$ sudo sed -i 's/GRUB_DISABLE_OS_PROBER=true/GRUB_DISABLE_OS_PROBER=false/' /etc/default/
grub
$ sudo os-prober
$ sudo grub-mkconfig -o /boot/grub/grub.cfg
```

EFI ВРУЧНУЮ — EFIBOOTMGR

Вкладка EFI — это интерфейс для `efibootmgr`. Всё, что она делает, соответствует одной команде:

```
# перечисляет все записи прошивки в текущем порядке загрузки
$ sudo efibootmgr -v
# переупорядочивает: ставит запись 0002 первой (= DEFAULT)
$ sudo efibootmgr -o 0002,0000,0001,0003
# разовая загрузка записи 0003 (= BOOT NEXT)
$ sudo efibootmgr -n 0003
# отменяет ожидающий BootNext
$ sudo efibootmgr -N
```

ОСИРОТЕВШИЕ ЗАГРУЗЧИКИ

Иногда установщик записывает свой загрузчик на EFI System Partition, не регистрируя для него запись прошивки — загрузчик существует на диске, но прошивка не знает, что он там есть. Панель Odyssey напрямую сканирует ESP и перечисляет их отдельно как **«ON DISK, NOT IN FIRMWARE»**, каждый с кнопкой REGISTER в один клик. Вручную эквивалент — `efibootmgr -c -d /dev/sdX -p N -L "Имя" -l '\EFI\имя\bootx64.efi'` — обратите внимание, что новосозданные записи ставятся самой прошивкой **первыми** в порядке загрузки, так что после этого заново примените DEFAULT к своей обычной записи, если не хотите, чтобы она заняла первое место.

ФАЙЛЫ, КОТОРЫЕ ЗАТРАГИВАЕТ ЭТА ПАНЕЛЬ

ФАЙЛ	РОЛЬ
`/etc/default/grub`	Командная строка ядра, таймаут, GRUB_DEFAULT, GRUB_DISABLE_OS_PROBER
`/boot/grub/grub.cfg`	Само сгенерированное меню загрузки — никогда не редактируйте вручную; перезаписывается grub-mkconfig
Переменные EFI (NVRAM)	Записи загрузки прошивки — это вообще не файл; читается и записывается полностью через efibootmgr

CHAPTER 3

Kernel

Четыре вкладки — **SYSCTL**, **CPU GOVERNOR**, **HUGEPAGES**, **MODULES** — плюс ряд предустановок в один клик над всеми четырьмя, который применяется ко всем сразу.

3.1 Визуальное руководство — предустановки

Вверху: версия вашего работающего ядра, активный планировщик и четыре кнопки предустановок — **DESKTOP**, **GAMING**, **SERVER**, **BALANCED**.

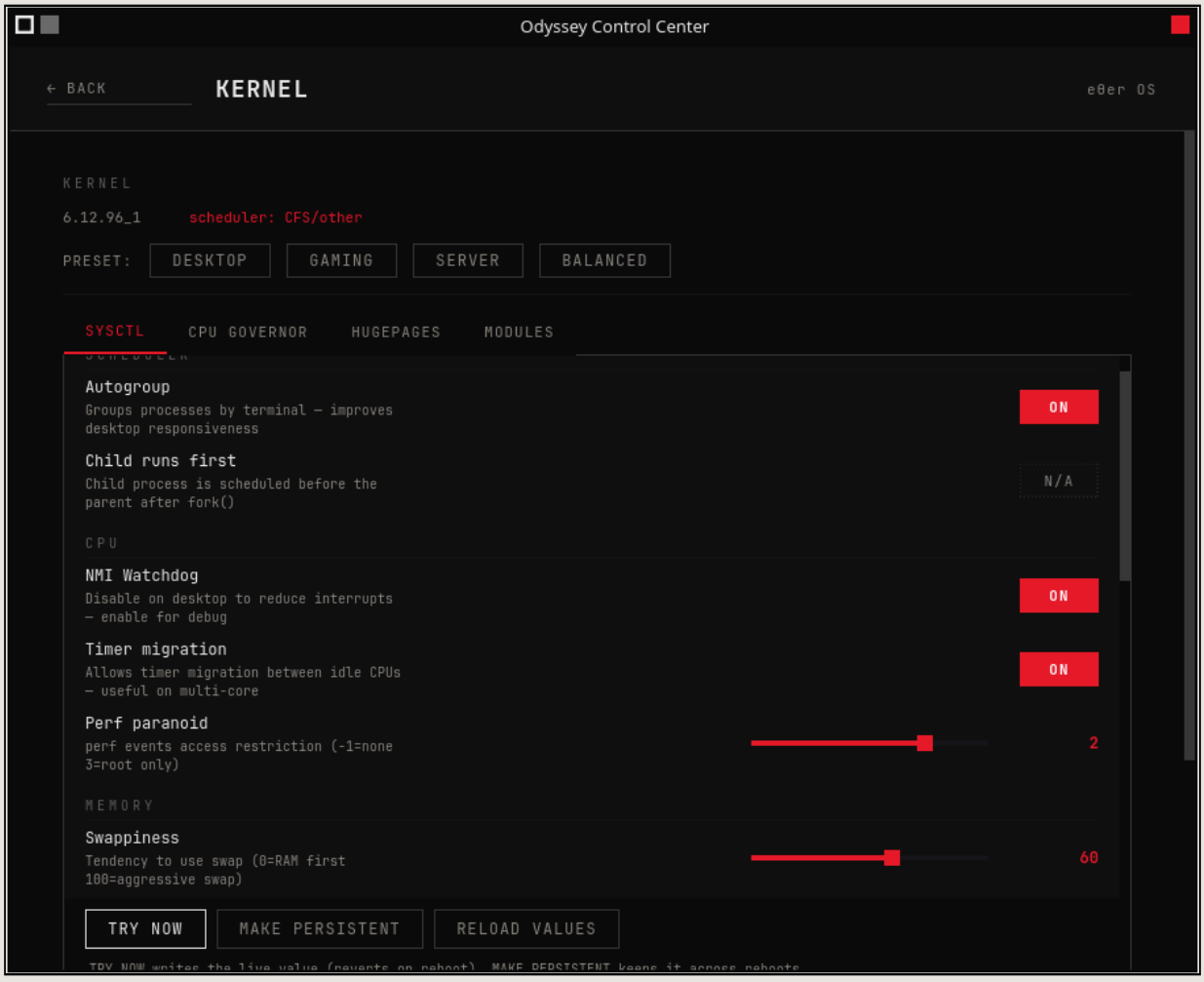

FIG · Только что открытая вкладка SYSCTL: ядро 6.12.96_1, планировщик CFS/other, четыре предустановки, и группы Scheduler/CPU/Memory ниже.

ПРЕДУСТАНОВКА	КОГДА ОНА ТОГО СТОИТ
DESKTOP	Повседневная машина — просмотр веб-страниц, офисная работа, лёгкая многозадачность. Более низкая swappiness, умеренное давление на кэш, миграция таймера включена для плавного энергопотребления в простое. Разумный вариант по умолчанию для большинства людей.
GAMING	Swappiness сведена почти к нулю (держите всё в RAM), миграция таймера отключена (ядра остаются выделенными вместо миграции работы), ограничения на события perf ослаблены для инструментов профилирования вроде MangoHud. Используйте, когда время кадра важнее энергопотребления в простое.
SERVER	Более высокая swappiness и терпимость к давлению на кэш, NMI watchdog включён, более строгие ограничения на указатели ядра/логи. Используйте на безголовой машине, которая ставит стабильность и безопасность выше интерактивной отзывчивости.
BALANCED	Компромисс между Desktop и Gaming — для машины, которая иногда рабочая станция, а иногда игровая сессия, и вы не хотите постоянно переключаться между предустановками.

WHAT	Kernel → нажмите DESKTOP, GAMING, SERVER или BALANCED.
WHY	Настраивать вручную десяток значений sysctl — это слишком много для того, кто просто хочет «хорошие настройки для игр» — предустановка это именно то, уже решённое.
RESULT	Каждое значение во вкладке SYSCTL переключается на цифры предустановки, готовые и видимые — вам всё равно нужны TRY NOW или MAKE PERSISTENT (ниже), чтобы действительно их применить.
RISK	Никакого — выбор предустановки только заполняет форму.

3.2 Визуальное руководство — SYSCTL, группа за группой

Прокрутите вниз, и каждый параметр сгруппирован под заголовком, у каждого есть описание простым языком прямо в панели — вам никогда не нужно знать, что означает `vm.compaction_proactiveness`, не прочитав об этом сначала там.

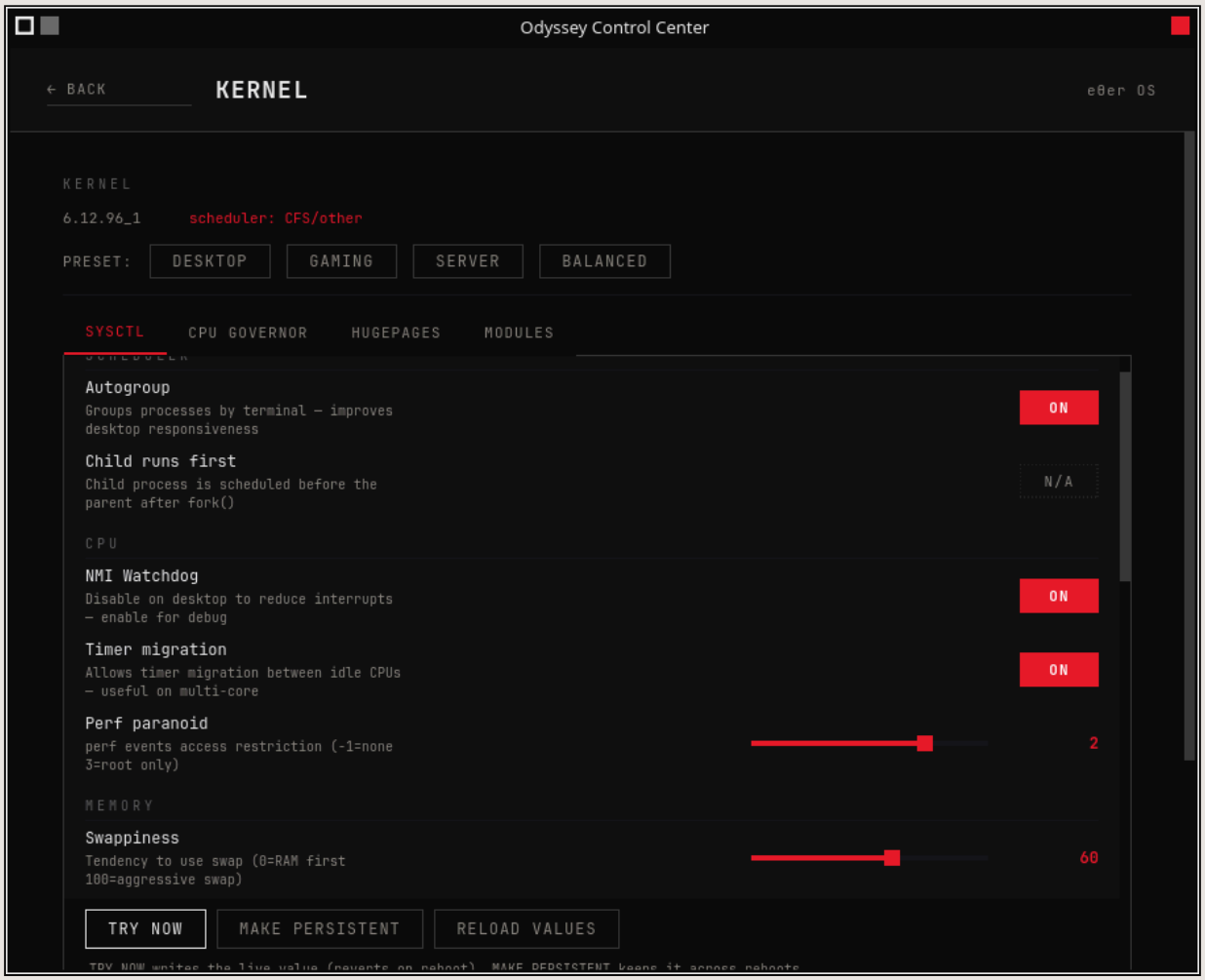

FIG · Группы Scheduler и CPU: Autogroup, Child runs first, NMI Watchdog, Timer migration, Perf paranoid — у каждого свой переключатель или ползунок и строка объяснения.

ГРУППА	ЧТО СОДЕРЖИТ
Scheduler	Как процессы группируются и приоритизируются
CPU	Watchdog, миграция таймера, ограничения на мониторинг производительности
Memory	Swappiness, давление на кэш, пороги записи «грязных» страниц
Network	TCP Fast Open, размер очереди backlog, лимиты буферов сокетов
Security	Скрыты ли указатели ядра и сообщения журнала от пользователей без привилегий

Две кнопки внизу применяют ваши изменения — и они намеренно разные:

WHAT	TRY NOW
WHY	Проверить значение с нулевым риском того, что оно переживёт неудачную перезагрузку.
RESULT	Записывается вживую через `sysctl` — действует немедленно, только для этой сессии.
RISK	Никакого — автоматически откатывается при перезагрузке.

WHAT	MAKE PERSISTENT
WHY	Вы проверили значение (или всю предустановку) и хотите, чтобы оно пережило перезагрузки.
RESULT	Записывается в `/etc/sysctl.d/99-odyssey.conf` **и** применяется вживую за тот же клик.
RISK	Низкий — это текстовый файл; удалите строку или примените заново, чтобы отменить.

RELOAD VALUES перечитывает то, что сейчас активно, отбрасывая всё введённое, но не применённое.

3.3 Визуальное руководство — CPU governor

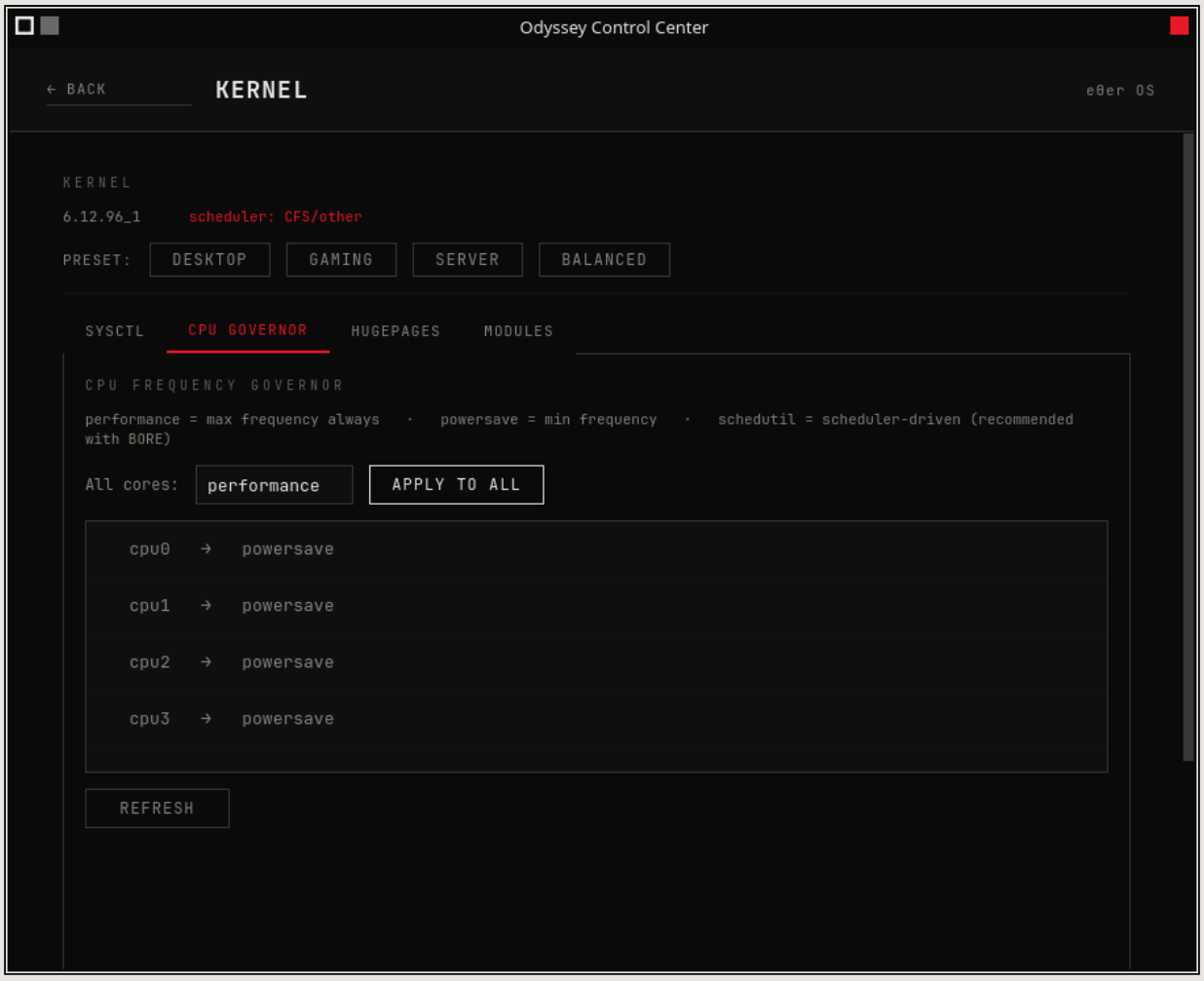

FIG · Вкладка CPU GOVERNOR: «All cores» установлено на performance с APPLY TO ALL, а список по ядрам ниже всё ещё показывает всё в powersave — несоответствие, ожидающее исправления.

Governor определяет, насколько агрессивно ваш CPU меняет тактовую частоту. Выпадающий список применяет governor ко всем ядрам сразу; список ниже показывает governor **вживую** для каждого отдельного ядра — так вы обнаруживаете ядро, застрявшее на неправильной настройке, именно как на скриншоте выше, где «All cores» говорит performance, но каждое ядро ниже всё ещё показывает powersave, что означает, что APPLY TO ALL ещё не нажимали.

GOVERNOR	ПОВЕДЕНИЕ
performance	Максимальная частота, всегда — самый отзывчивый, самое высокое потребление
powersave	Минимальная частота, всегда — самый тихий, наименее отзывчивый
schedutil	Частота управляется собственным видением нагрузки планировщиком — рекомендуется с планировщиком BORE

В выпадающем списке появляются только те governor'ы, которые действительно предоставляют ваши конкретные CPU и ядро.

3.4 Визуальное руководство — hugepages

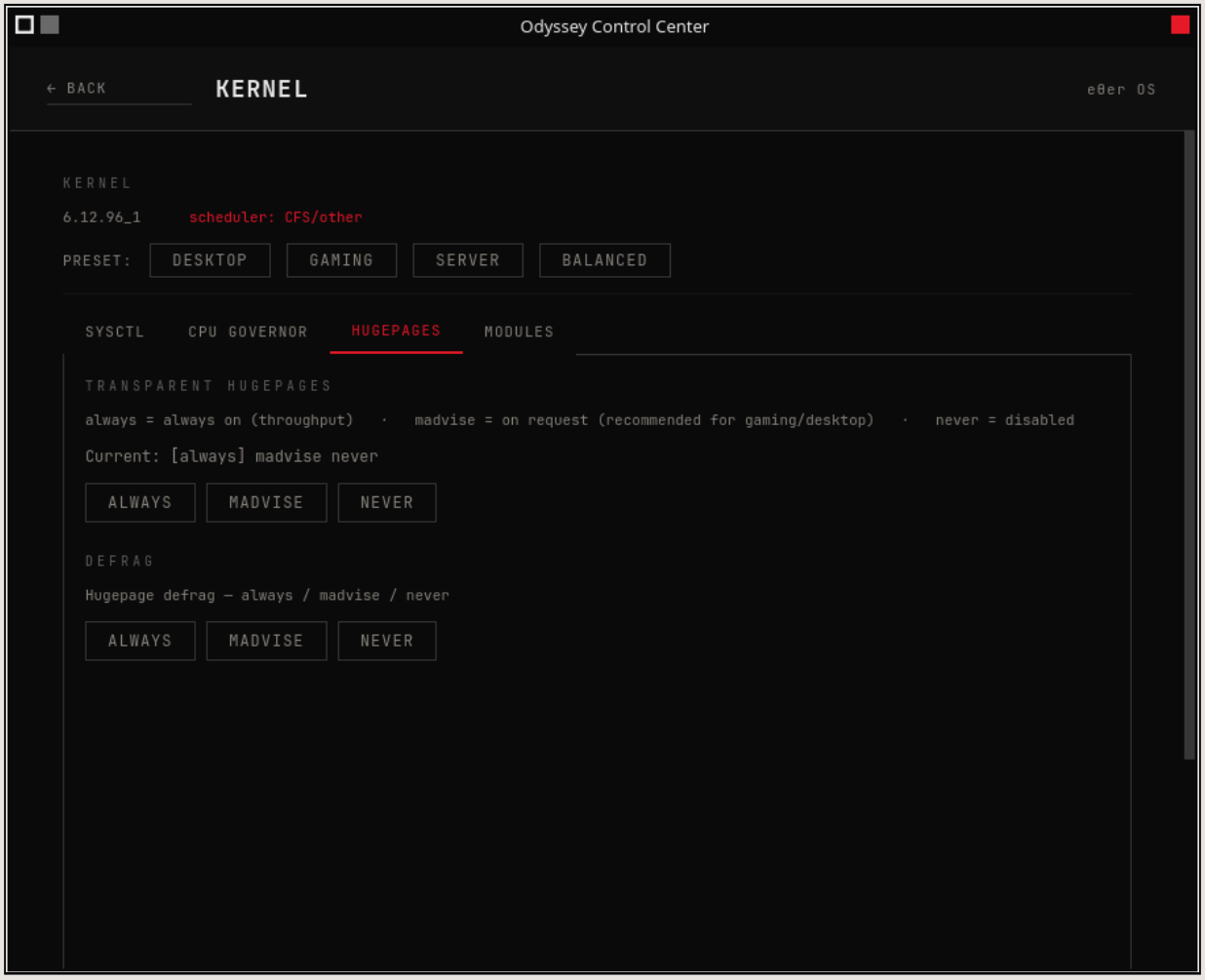

FIG · Вкладка HUGEPAGES: Transparent Hugepages показывает «Current: [always] madvise never» с выделенным ALWAYS, и тот же ряд из трёх кнопок снова для Defrag ниже.

Две независимые настройки, каждая — ряд из трёх кнопок, применяемых мгновенно по клику — текущее значение вживую всегда показано в квадратных скобках над кнопками, как `[always] madvise never` на скриншоте.

НАСТРОЙКА	ЧТО ОНА КОНТРОЛИРУЕТ
Transparent hugepages	ALWAYS = всегда включено везде (максимальная пропускная способность); MADVISE = только когда приложение явно этого просит (рекомендуется для игр/десктопа); NEVER = полностью отключено
Defrag	Насколько агрессивно ядро уплотняет память, чтобы **сформировать** hugepages по запросу — те же три варианта, независимо от настройки выше

3.5 Визуальное руководство — modules

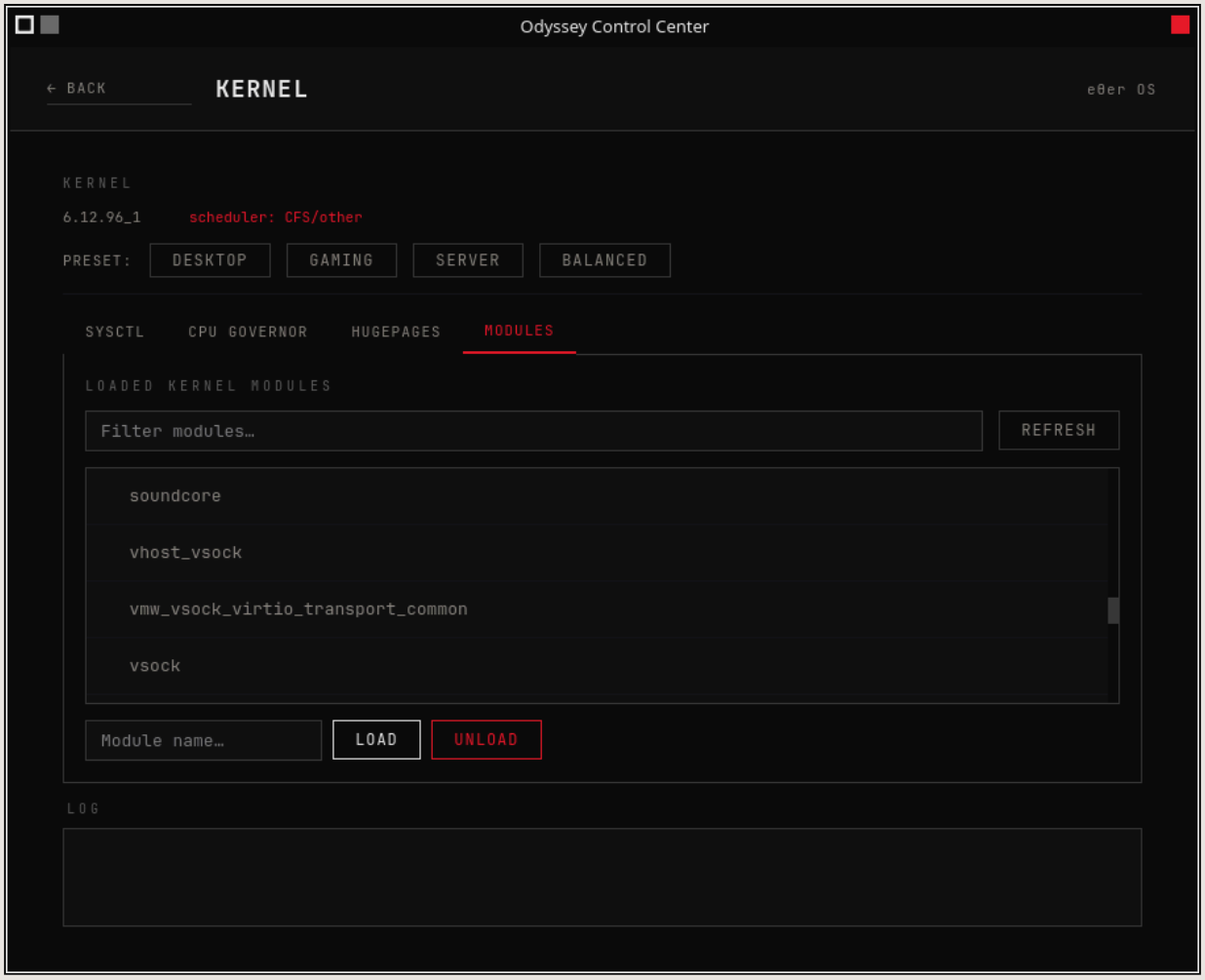

FIG · Вкладка MODULES: поле фильтра, список вживую (soundcore, vhost_vsock, vsock…), и ряд LOAD/UNLOAD внизу для модуля, введённого по имени.

Живой и фильтруемый список каждого загруженного в данный момент модуля ядра — введите текст в фильтр, чтобы сузить список, REFRESH для повторного чтения. В отличие от трёх других вкладок, эта позволяет действовать напрямую: введите имя модуля и нажмите **LOAD** или **UNLOAD**.

WHAT	Kernel → вкладка MODULES → введите имя модуля → LOAD или UNLOAD.
WHY	Иногда нужен определённый драйвер или функциональный модуль активным (или удалённым) без перезагрузки — протестировать оборудование, освободить устройство, которое удерживает другой модуль.
RESULT	Модуль немедленно вставляется в работающее ядро или удаляется из него.
RISK	Умеренный, конкретно для UNLOAD — удаление модуля, от которого зависит что-то другое, может дестабилизировать то, что его использовало (видеодрайвер, файловую систему). LOAD в целом безопасен; ядро сразу отказывается загружать несовместимый модуль, а не применяет его наполовину.

▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.

3.6 Advanced

ТОЧНЫЕ КЛЮЧИ SYSCTL

КЛЮЧ	ПОДПИСЬ В ПАНЕЛИ
kernel.sched_autogroup_enabled	Autogroup
kernel.sched_child_runs_first	Child runs first
kernel.nmi_watchdog	NMI Watchdog
kernel.timer_migration	Timer migration
kernel.perf_event_paranoid	Perf paranoid
vm.swappiness	Swappiness
vm.vfs_cache_pressure	VFS cache pressure
vm.dirty_ratio / vm.dirty_background_ratio	Dirty ratio % / Dirty background ratio %
vm.compaction_proactiveness	Compaction proactiveness
vm.watermark_boost_factor	Watermark boost
net.ipv4.tcp_fastopen	TCP Fast Open
net.core.netdev_max_backlog	Netdev max backlog
net.core.rmem_max	Socket rmem max
kernel.kptr_restrict / kernel.dmesg_restrict	kptr restrict / dmesg restrict

ЗНАЧЕНИЯ ПРЕДУСТАНОВОК, DESKTOP VS GAMING

```
# DESKTOP
$ sysctl vm.swappiness=10 vm.vfs_cache_pressure=50 kernel.timer_migration=1
# GAMING
$ sysctl vm.swappiness=1 vm.vfs_cache_pressure=50 kernel.timer_migration=0
# полные таблицы предустановок можно прочитать прямо в panel_kernel.py —
# откройте исходный код Control Center, если хотите увидеть каждое поле
```

ВРУЧНУЮ, БЕЗ ПАНЕЛИ

```
# вживую, временно
$ sudo sysctl vm.swappiness=10
# постоянно — тот же файл, что пишет панель
$ echo “vm.swappiness=10” | sudo tee -a /etc/sysctl.d/99-odyssey.conf
$ sudo sysctl -system
# CPU governor, по ядрам
$ echo schedutil | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
# hugepages
$ echo madvise | sudo tee /sys/kernel/mm/transparent_hugepage/enabled
$ echo madvise | sudo tee /sys/kernel/mm/transparent_hugepage/defrag
# модули
$ sudo modprobe MODULE_NAME
$ sudo modprobe -r MODULE_NAME
```

ПОЧЕМУ TRY NOW И MAKE PERSISTENT — РАЗДЕЛЬНЫЕ ДЕЙСТВИЯ

Записи `sysctl` по своей природе всегда только вживую — у ядра нет встроенной постоянной памяти. MAKE PERSISTENT — это Odyssey, выбирающий единый канонический файл, `/etc/sysctl.d/99-odyssey.conf`, и выполняющий и запись вживую, и запись в файл вместе, чтобы они никогда не расходились. Редактирование этого файла напрямую и запуск `sysctl --system` после этого функционально идентичны нажатию кнопки.

CHAPTER 4

Memory & Swap

Один экран, три раздела: сводка только для чтения, swappiness и ZRAM.

4.1 Визуальное руководство

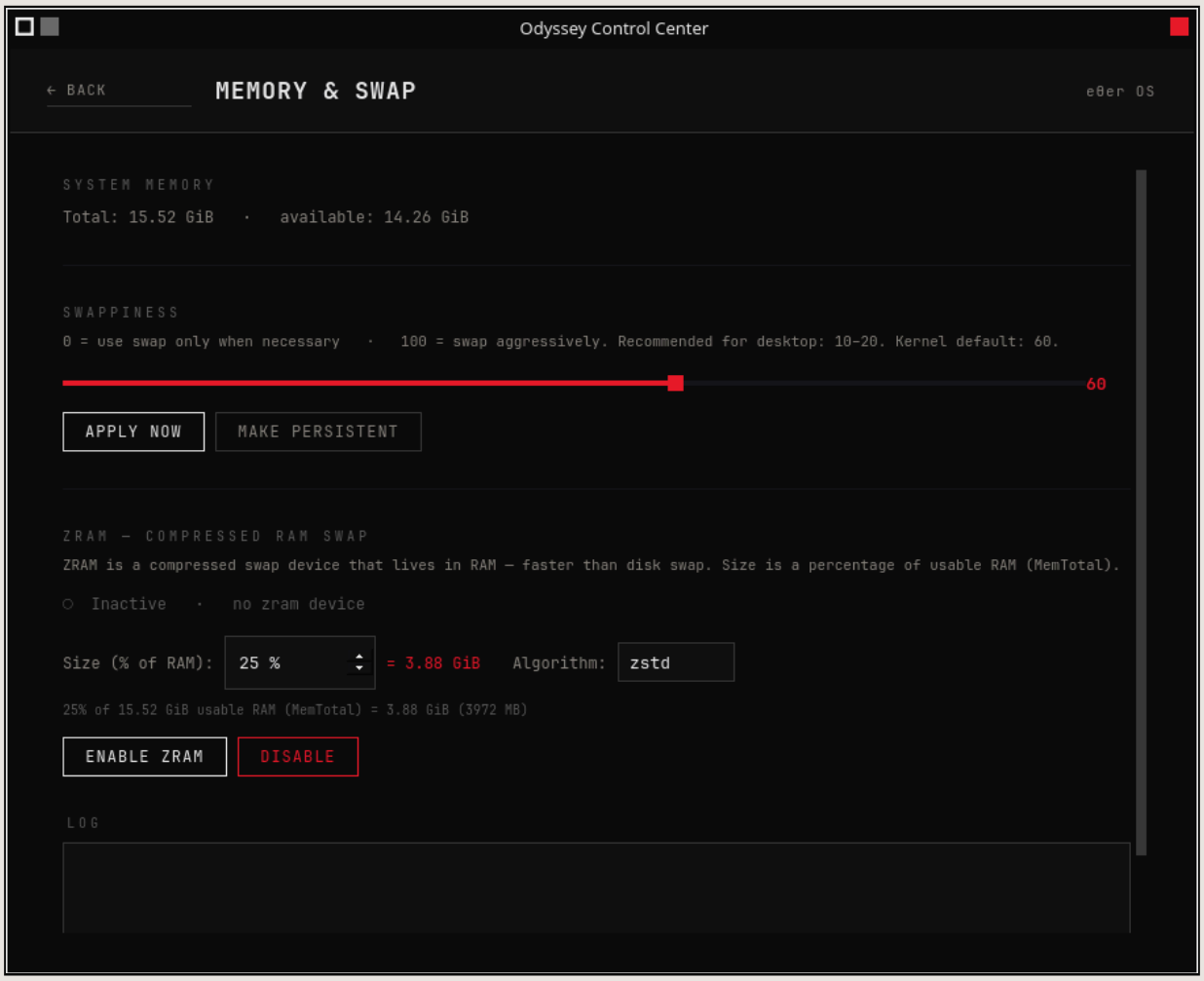

FIG · Полная панель: System memory (всего 15,52 ГиБ, доступно 14,26 ГиБ), ползунок Swappiness на 60 с APPLY NOW / MAKE PERSISTENT, и ZRAM ниже, показывающий «Inactive · no zram device» с размером, алгоритмом и ENABLE/DISABLE.

SYSTEM MEMORY

Просто сводка вверху — общая RAM и то, что сейчас доступно. Никаких элементов управления; это контекст для двух разделов ниже.

SWAPPINESS

Один ползунок от 0 до 100. 0 означает «использовать swap только тогда, когда действительно нет другого выбора»; 100 означает «использовать swap агрессивно, задолго до заполнения RAM». Значение по умолчанию в ядре — 60 (то, что показывает скриншот); собственная подсказка панели предлагает 10–20 для десктопа — то же число, которое задаёт предустановка DESKTOP панели Kernel.

WHAT	Перетащите ползунок → APPLY NOW, чтобы проверить, MAKE PERSISTENT, когда будете довольны.
WHY	Более низкое значение удерживает то, что вы активно используете, в быстрой RAM, а не перемещает в более медленный swap — обычно то, что вам нужно при достаточном объёме установленной памяти.
RESULT	APPLY NOW меняет его вживую, немедленно, только для этой сессии. MAKE PERSISTENT записывает его в конфигурационный файл, чтобы оно пережило и перезагрузку.
RISK	Очень низкий. Чрезвычайно низкое значение на машине, которой действительно не хватает RAM, может сделать её более склонной резко исчерпать память при высокой нагрузке вместо плавного свопинга — если так случится, поднимите значение обратно.

ZRAM — СЖАТЫЙ SWAP В RAM

ZRAM — это устройство подкачки, которое живёт **внутри** самой RAM, в сжатом виде — заметно быстрее, чем swap на диске, потому что диск вообще не задействован. Размер — это процент от вашей общей доступной RAM, а не фиксированное число, поэтому настройка разумно масштабируется на разных машинах; показание вживую рядом с полем размера переводит процент в ГиБ для вас (на скриншоте: 25% = 3,88 ГиБ в системе с 15,52 ГиБ).

WHAT	Задайте процент размера и выберите алгоритм (zstd, lz4, lzo или deflate) → ENABLE ZRAM.
WHY	Сжатый swap в RAM поглощает давление на память с намного меньшим влиянием на производительность, чем swap на диске — полезно на любой машине, особенно с ограниченной RAM.
RESULT	Устройство `zram0` создаётся и активируется с выбранным размером и алгоритмом; строка статуса меняется с «Inactive» на активное.
RISK	Низкий. zstd (по умолчанию) хорошо балансирует степень сжатия и затраты CPU для большинства машин. DISABLE отключает его чисто в любой момент.

▸ **ADVANCED**

Past this point: config file paths, raw commands, and what the panel is doing underneath.

4.2 Advanced

ЧТО НА САМОМ ДЕЛЕ ОЗНАЧАЕТ «ЗАВИСИТ ОТ ОБОРУДОВАНИЯ» ЗДЕСЬ

Обе настройки взаимодействуют с вашим конкретным оборудованием, а не просто по принципу «больше RAM = лучше»:

- **Быстрый NVMe SSD**: swappiness в диапазоне 10–20 стоит очень мало, даже если происходит свопинг — диск достаточно быстр, чтобы это было едва заметно. ZRAM всё равно выигрывает по задержке и износу SSD, но разрыв меньше, чем с механическим диском.
- **Механический HDD или eMMC как swap**: держите swappiness низкой (0–10) — swap на диске на медленных носителях вызывает подтормаживание, которое называют «thrashing». Именно здесь ZRAM даёт гораздо больший выигрыш.
- **Запас CPU**: ZRAM обменивает дисковый I/O на циклы CPU, потраченные на сжатие и распаковку. На машине, ограниченной по CPU (старое оборудование или уже загруженное во время игры), `lz4` сжимает быстрее, чем `zstd`, но с худшим соотношением — разумный компромисс, если время CPU — более дефицитный ресурс. На всём современном лучшее соотношение `zstd` стоит своей немного более высокой цены.
- **Запас RAM**: большой процент ZRAM (например, 50%) на машине только с 8 ГиБ в сумме оставляет меньше реальной RAM для приложений, поскольку сам ZRAM потребляет реальную память для хранения сжатых страниц. При 16 ГиБ и более 25% — комфортное значение по умолчанию с запасом.

ВРУЧНУЮ

```
# swappiness, вживую
$ sudo sysctl vm.swappiness=15
# swappiness, постоянно — тот же файл, что пишет панель
$ echo “vm.swappiness=15” | sudo tee /etc/sysctl.d/99-odyssey.conf
# zram, ручная настройка
$ sudo modprobe zram
$ echo zstd | sudo tee /sys/block/zram0/comp_algorithm
$ echo 4G | sudo tee /sys/block/zram0/disksize
$ sudo mkswap /dev/zram0
$ sudo swapon /dev/zram0 -p 100
# проверьте, что оно активно
$ swapon -show
```

ПОЧЕМУ ПАНЕЛЬ СБРАСЫВАЕТ ZRAM0, А НЕ ИЗМЕНЯЕТ ЕГО РАЗМЕР ВЖИВУЮ

Размер устройства ZRAM нельзя изменить, пока оно активно и используется как swap — ядро требует сначала его отключить и сбросить. Кнопка ENABLE ZRAM панели выполняет эту последовательность автоматически (swapoff, сброс, переконфигурация, swapon), поэтому изменение размера или алгоритма на уже активном устройстве всё равно выглядит как одно чистое действие, а не как ошибка.

CHAPTER 5

Firewall

У панели Firewall есть собственная выделенная глава, во всей глубине, в **Security and Hardening Guide** — набор правил, объяснённый строка за строкой, все 21 предустановка с их индивидуальными рисками, редактор необработанной конфигурации, и эквиваленты командной строки. Эта глава — визуально направленная версия.

5.1 Визуальное руководство

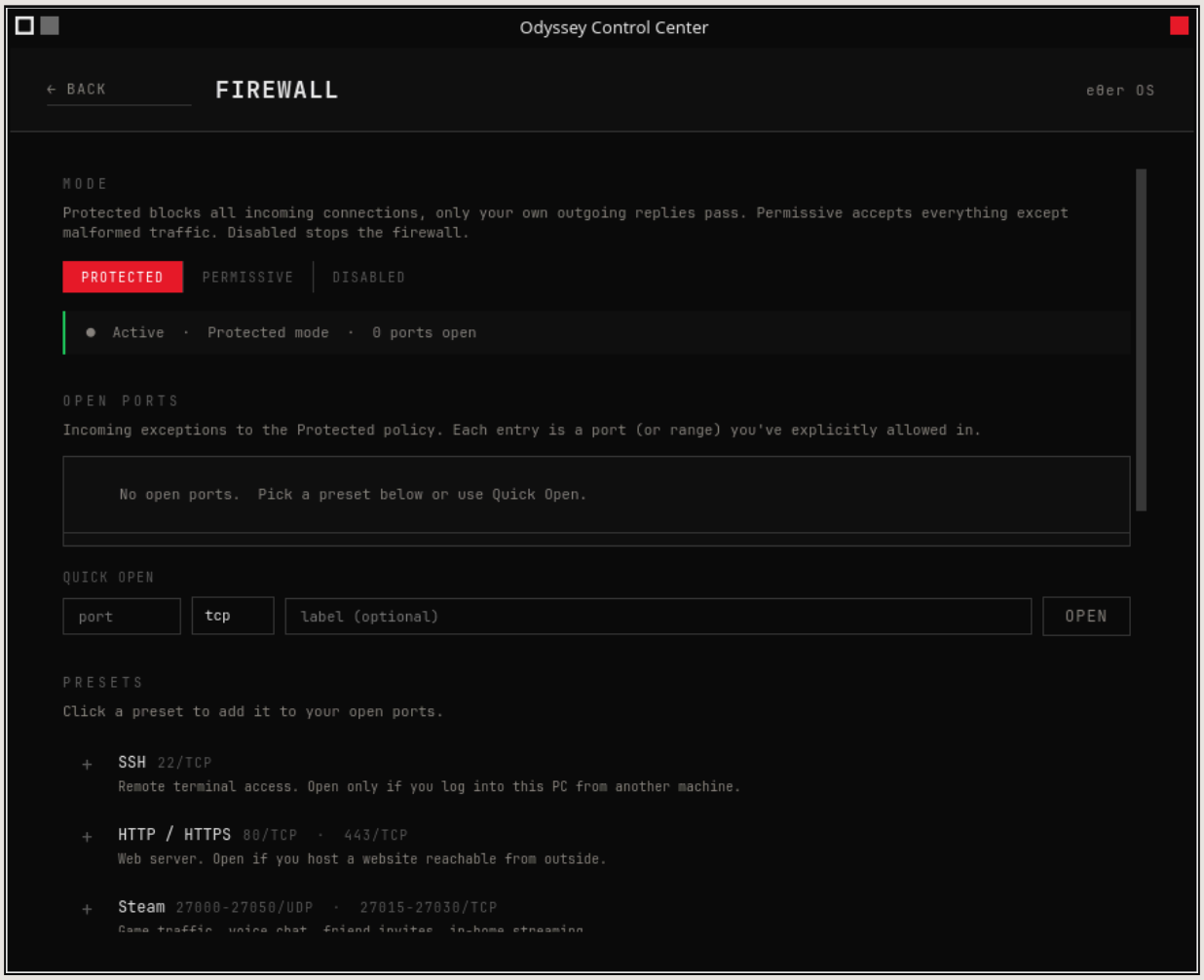

FIG · Режим PROTECTED активен, «0 ports open», поля Quick Open, и плитки предустановок, начинающиеся с SSH, HTTP/HTTPS и Steam.

MODE

Три кнопки вверху, и панель объясняет каждую в одной строке прямо над ними:

РЕЖИМ	ЧТО ОН ДЕЛАЕТ
PROTECTED	Значение по умолчанию. Блокирует все входящие соединения; проходят только ваши собственные исходящие ответы.
PERMISSIVE	Принимает всё, кроме искажённого трафика — диагностическая настройка для вопроса «это файрвол ломает работу?», а не место, где стоит оставаться.
DISABLED	Полностью останавливает файрвол.

Строка статуса ниже подтверждает, что на самом деле активно — на скриншоте **«Active · Protected mode · 0 ports open»**.

OPEN PORTS И ПРЕДУСТАНОВКИ

OPEN PORTS перечисляет ваши явные исключения — по умолчанию пусто, как показано. **QUICK OPEN** принимает порт, протокол и опциональную метку напрямую, для всего, чего нет в списке предустановок. **PRESETS** ниже предлагает восемь распространённых служб по одному клику каждая, с строкой, объясняющей, когда они действительно нужны (SSH: «удалённый доступ к терминалу, открывайте только если заходите на этот ПК с другой машины»), плюс ещё тринадцать под раскрывающимся «MORE PRESETS».

WHAT	Firewall → нажмите на плитку предустановки, или заполните QUICK OPEN → SAVE & APPLY.
WHY	Вы запускаете что-то — игровой сервер, медиасервер, сервер разработки — что должно быть доступно с другой машины.
RESULT	Этот порт открывается; всё остальное остаётся закрытым политикой PROTECTED.
RISK	Полностью зависит от предустановки — сверьтесь с заметками о риске для каждой предустановки в Security and Hardening Guide, прежде чем открывать что-либо помимо SSH или HTTP/HTTPS на машине, доступной из интернета.

Ничего не действует, пока вы не нажмёте **SAVE & APPLY** — изменения сначала ставятся в ожидание, чтобы вы могли точно проверить, что собираетесь открыть, прежде чем подтвердить.

```
▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.
```

5.2 Advanced

Полный технический справочник — набор правил по умолчанию из `nftables.conf`, объяснённый правило за правилом, каждая из 21 предустановки с её конкретной уязвимостью и способом её смягчения, рецепты для ограничения доступа только локальной сетью, эквиваленты командной строки `nft`, и решение проблемы с заблокированной снаружи сессией SSH — находится в **Главе 2 Security and Hardening Guide** (`/usr/share/doc/odyssey/` на вашей установленной системе). Это руководство намеренно не дублирует её; считайте ту главу продолжением этого раздела.

```
# команду, которую стоит знать прямо сейчас:
$ sudo nft list ruleset
# показывает точно, что загружено в ядре, независимо от панели
```

CHAPTER 6

AppArmor

Пять вкладок: **STATUS**, **PROFILES**, **COMMUNITY**, **VIOLATIONS**, **EDITOR**. Та же ссылка, что и у Firewall: подробный справочник — что такое обязательный контроль доступа на самом деле, полный жизненный цикл ограничения, практический пример — находится в Security and Hardening Guide. Вот визуальное руководство.

6.1 Визуальное руководство — STATUS

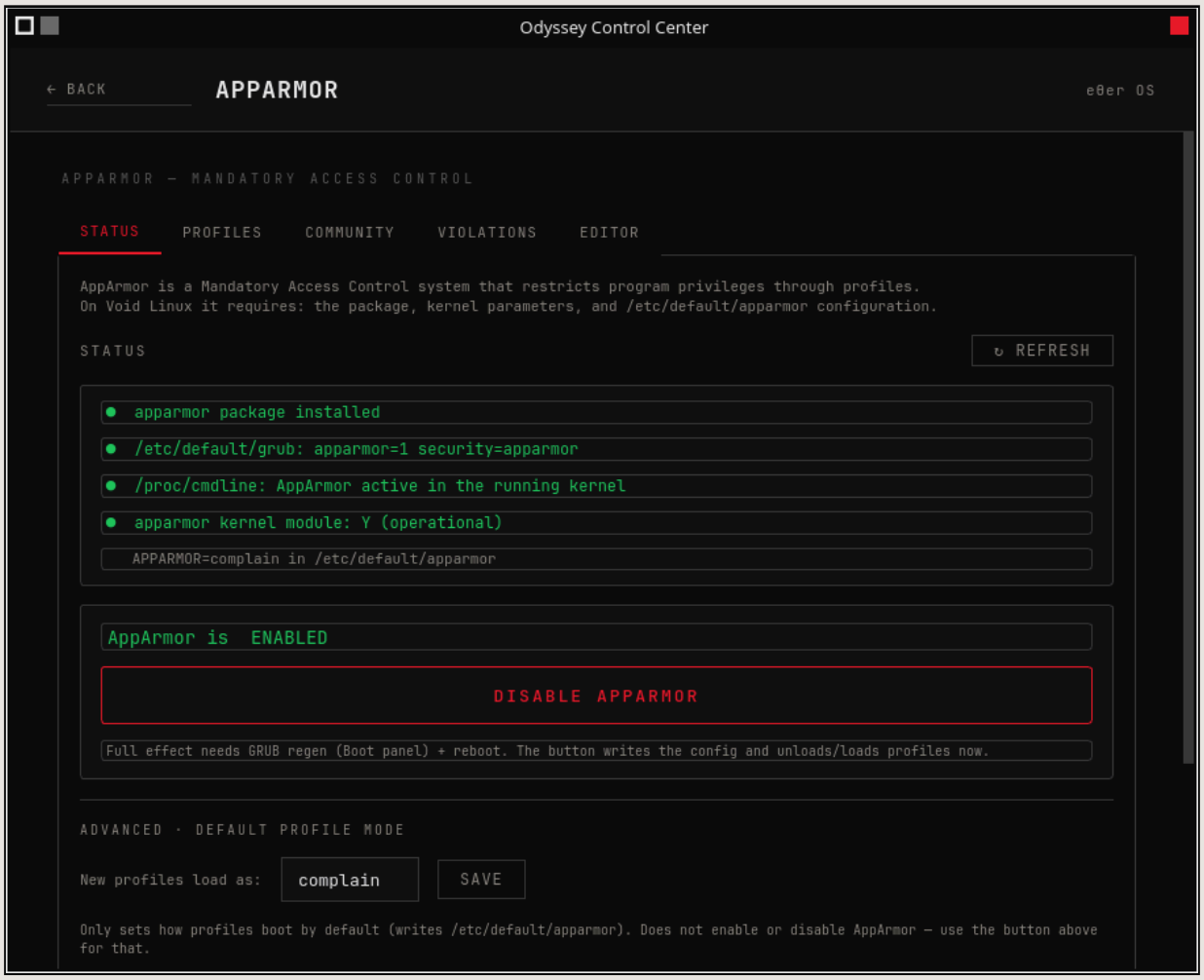

FIG · Вкладка STATUS: четыре зелёные галочки (пакет установлен, параметры GRUB, работающее ядро, модуль ядра), «AppArmor is ENABLED» с красной кнопкой DISABLE APPARMOR, и режим профиля по умолчанию, установленный на complain.

Четыре проверки подтверждают, что фреймворк действительно работает, сверху вниз: пакет `apparmor` установлен, параметры ядра в GRUB заданы, **работающее** ядро действительно загрузилось с ними (это та проверка, что может отставать, если вы не перезагружались после их

включения), и сам модуль ядра работоспособен. Ниже: большая кнопка включения/выключения и **ADVANCED · DEFAULT PROFILE MODE** — запускаются ли свежезагруженные профили в режиме complain или enforce.

WHAT	AppArmor → STATUS → убедитесь, что все четыре проверки зелёные.
WHY	Если какая-то проверка красная или жёлтая, ничто другое в этой панели не может работать правильно — это всегда первое, на что нужно смотреть, когда что-то с AppArmor кажется неправильным.
RESULT	Уверенность, что фреймворк действительно активен, а не просто установлен.
RISK	Никакого — проверка только для чтения.

6.2 Визуальное руководство — PROFILES

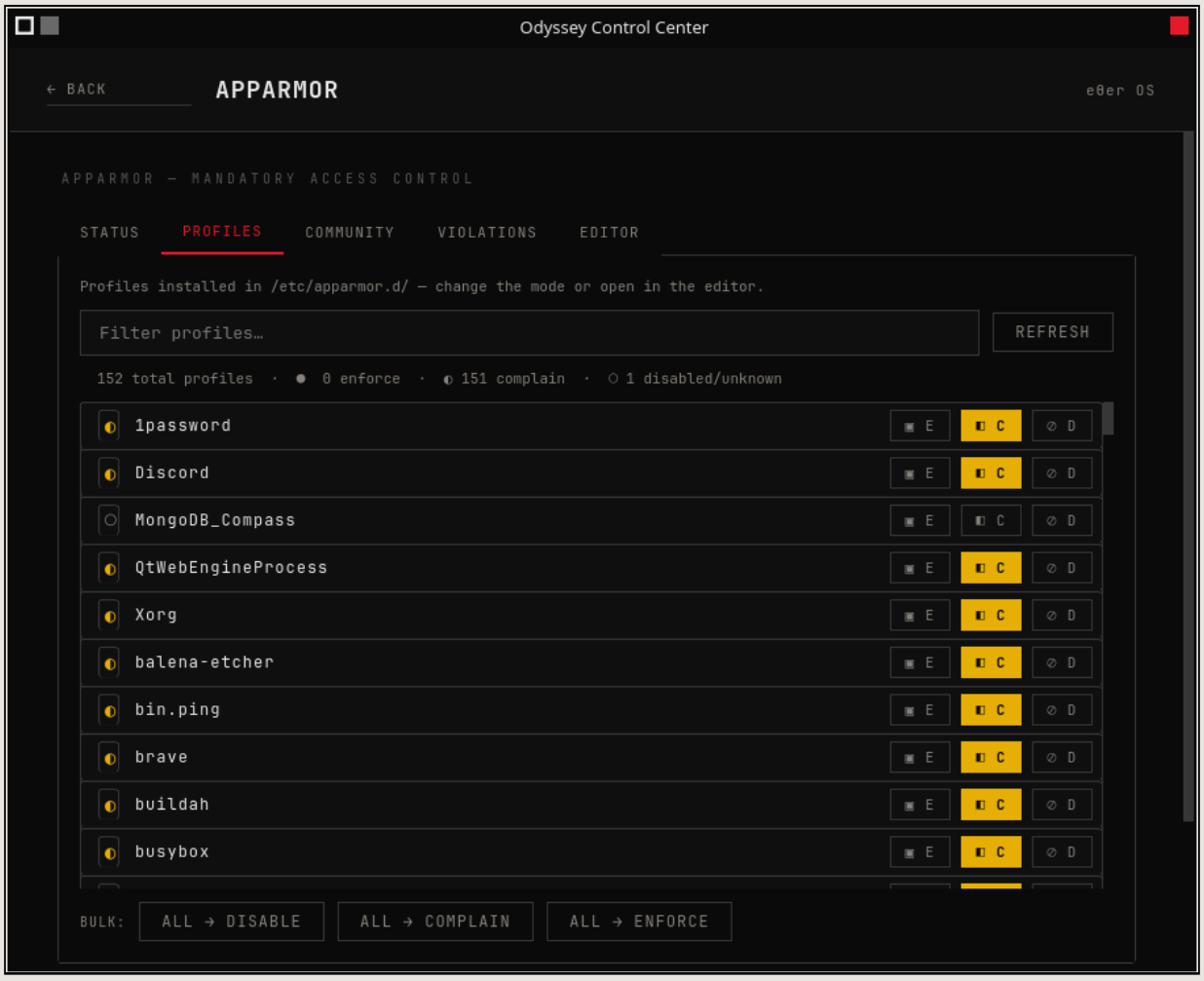

FIG · Вкладка PROFILES: всего 152 профиля, 0 enforce, 151 complain, 1 disabled/unknown — фильтруемый список с 1password, Discord, MongoDB_Compass и другими, каждый с кнопками E / C / D, и внизу ALL→DISABLE / ALL→COMPLAIN / ALL→ENFORCE.

Каждый загруженный профиль — своя строка, с полем фильтра и счётчиком вверху (на скриншоте: 152 профиля, все, кроме одного, в режиме complain — значение по умолчанию в Odyssey). Три кнопки на строку:

КНОПКА	ЭФФЕКТ
E	Переключает этот профиль в enforce — нарушения блокируются, а не только регистрируются
C	Переключает в complain — регистрирует нарушения, ничего не блокирует
D	Полностью отключает профиль — программа работает без ограничений

Три кнопки **BULK** внизу применяют одно и то же переключение ко всем профилям в списке сразу — полезно сразу после установки партии из COMMUNITY, но относитесь к ALL→ENFORCE с особой осторожностью; см. блок действия в следующем разделе.

6.3 Визуальное руководство — COMMUNITY

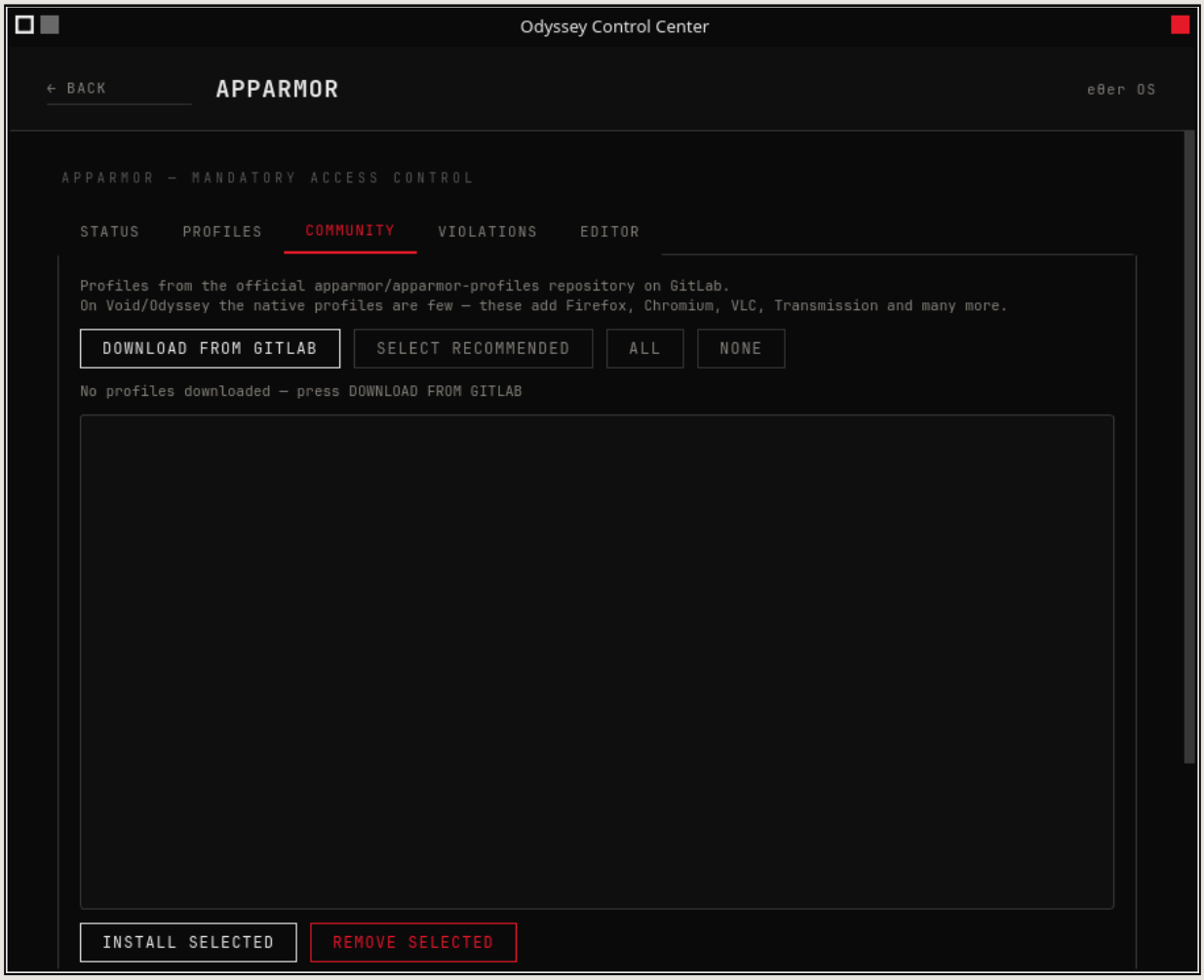

FIG · Вкладка COMMUNITY: «No profiles downloaded — press DOWNLOAD FROM GITLAB», с кнопками DOWNLOAD FROM GITLAB, SELECT RECOMMENDED, ALL и NONE, и INSTALL SELECTED / REMOVE SELECTED внизу.

AppArmor из коробки ограничивает в основном системные компоненты — настольные приложения, о которых вы действительно беспокоитесь (браузеры, чат-приложения, медиаплееры), нуждаются в профилях от сообщества. Эта вкладка получает их напрямую из официального репозитория `apparmor/apparmor-profiles` на GitLab.

1 **DOWNLOAD FROM GITLAB**

Получает текущий список доступных профилей — ничего ещё не устанавливается, это только заполняет список.

2 **SELECT RECOMMENDED (или ALL / NONE)**

Отмечает разумный набор по умолчанию для десктопа — браузеры семейства Firefox, Chromium, VLC, Transmission и подобные.

3 **INSTALL SELECTED**

Устанавливает отмеченные профили — они загружаются в режиме complain, как и всё остальное, готовые к циклу наблюдай-затем-применяй ниже.

WHAT	AppArmor → COMMUNITY → DOWNLOAD FROM GITLAB → SELECT RECOMMENDED (или выберите по отдельности) → INSTALL SELECTED.
WHY	Приложения, наиболее подверженные ненадёжному вводу — браузер, рендерящий вредоносную страницу, медиаплеер, открывающий сфабрикованный файл — это именно те, которые профиль защищает лучше всего.
RESULT	Новые профили появляются в PROFILES, в режиме complain, регистрируя то, что они бы заблокировали, но пока ничего не блокируя.
RISK	Никакого при установке — ничего не применяется, пока вы не решите в PROFILES.

6.4 Визуальное руководство — VIOLATIONS

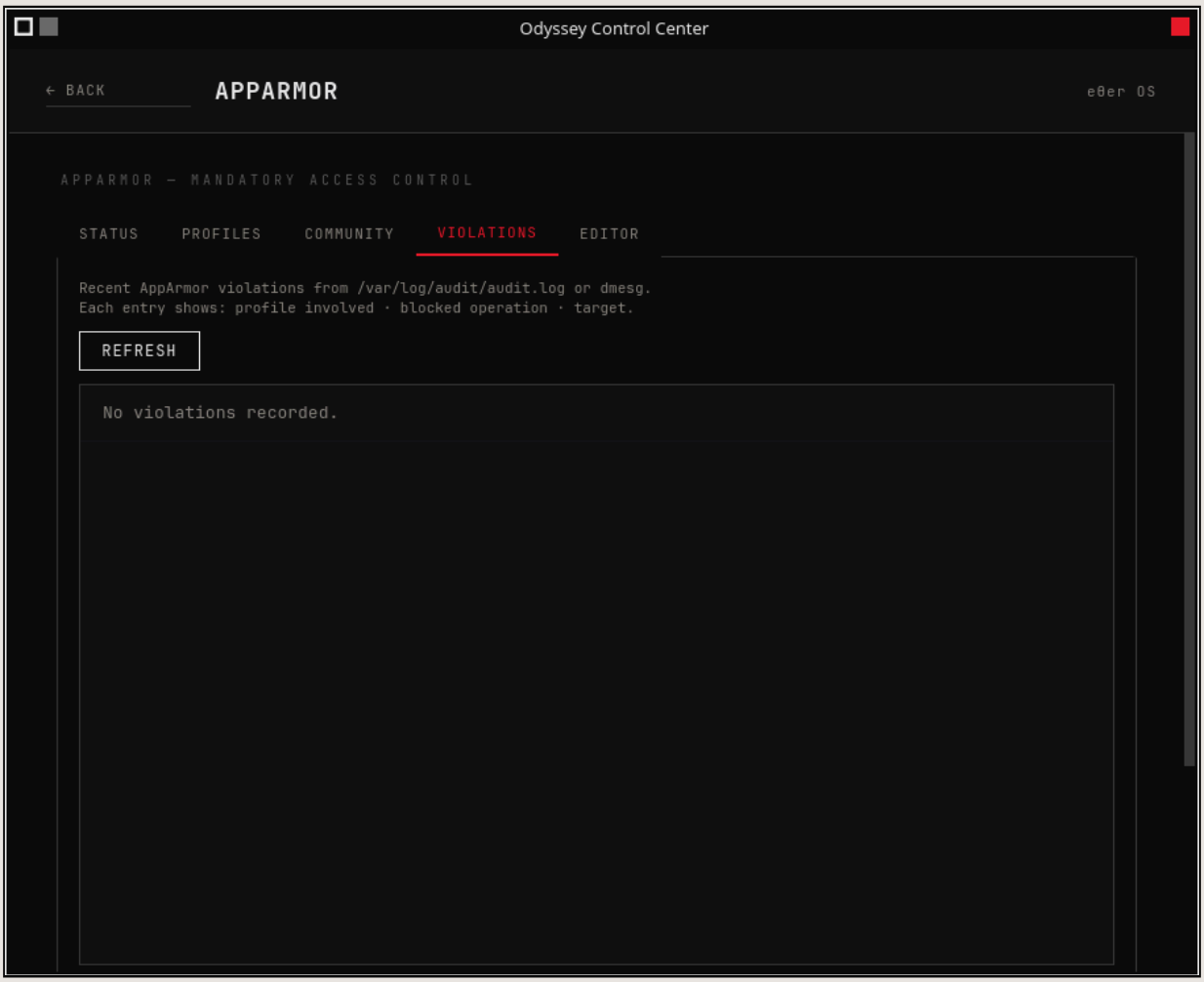

FIG · Вкладка VIOLATIONS: «No violations recorded», с кнопкой REFRESH и заметкой, что записи берутся из /var/log/audit/audit.log или dmesg.

Эта вкладка читает журнал аудита AppArmor системы и показывает, что профиль **заблокировал бы** (в режиме complain) или **заблокировал** (в режиме enforce) — каждая запись указывает профиль, заблокированную операцию и целевой файл или ресурс.

> **ПУСТОЙ СПИСОК — ОЖИДАЕМАЯ КАРТИНА НА ЗДОРОВОЙ СИСТЕМЕ**
>
> Скриншот показывает именно то, что вы обычно будете видеть: никаких нарушений. Это не ошибка и не признак того, что панель не работает — это значит, что каждая профилированная программа оставалась в допустимых границах с момента последней проверки журнала. Нарушения появляются здесь в момент, когда что-то профилированное пытается сделать то, что его профиль не разрешает; идите использовать соответствующую программу, вернитесь и нажмите REFRESH.

Как только записи появляются, двойной клик по одной из них — быстрый путь в EDITOR с уже открытым нужным профилем — см. ниже.

6.5 Визуальное руководство — EDITOR

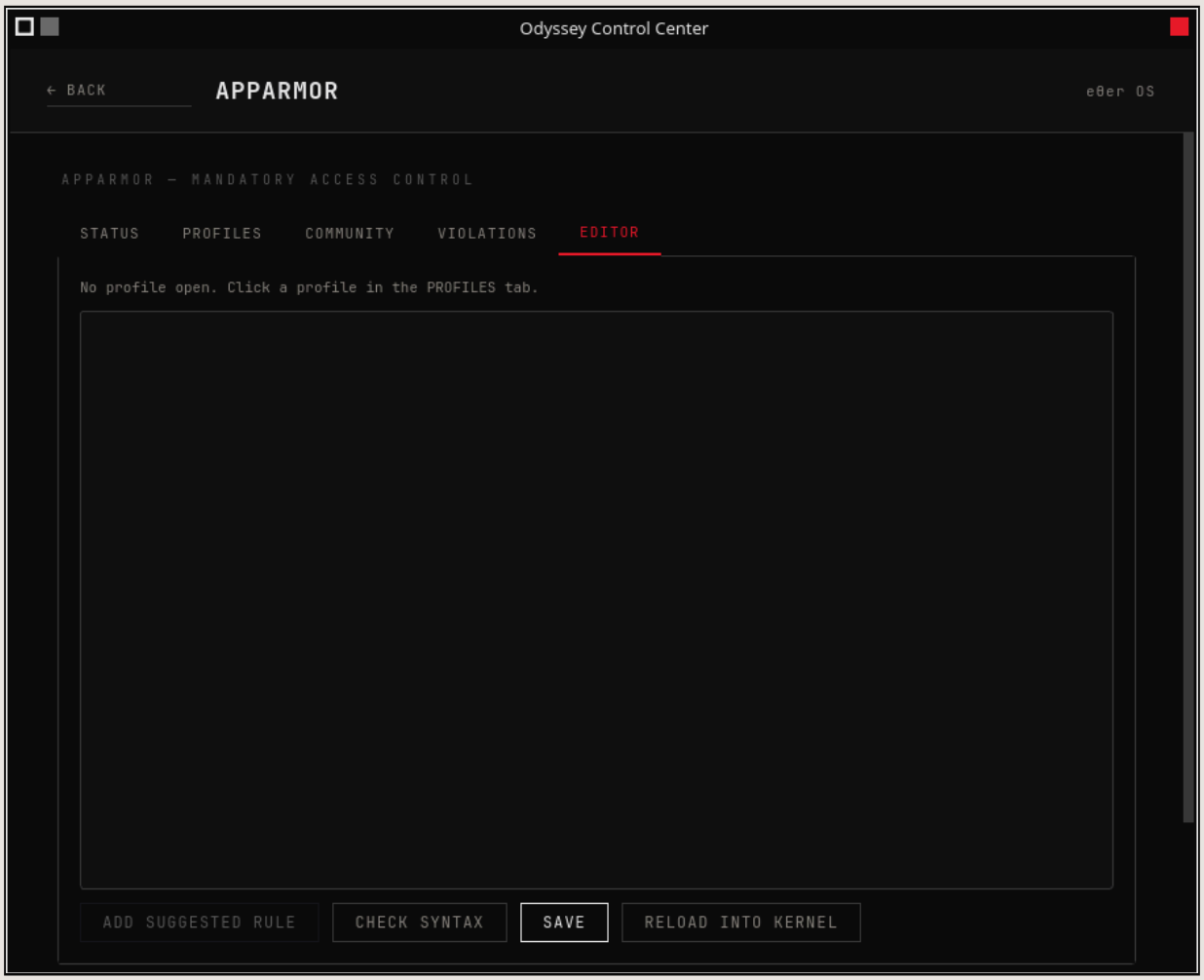

FIG · Вкладка EDITOR: «No profile open. Click a profile in the PROFILES tab», пустая текстовая область и четыре кнопки — ADD SUGGESTED RULE, CHECK SYNTAX, SAVE, RELOAD INTO KERNEL — все, кроме SAVE, отключены.

Редактор намеренно открывается **пустым**, как показано — он всегда редактирует только один профиль за раз, и вы выбираете, какой, из PROFILES (клик по строке) или из VIOLATIONS (двойной клик по записи). Это предотвращает лёгкую ошибку случайного редактирования не того файла.

КНОПКА	ЧТО ОНА ДЕЛАЕТ
ADD SUGGESTED RULE	Активна только если вы пришли из записи VIOLATIONS — вставляет точную строку правила, которая разрешила бы зарегистрированное действие
CHECK SYNTAX	Проверяет профиль без его загрузки — ловит опечатку прежде, чем она станет проблемой
SAVE	Записывает ваши изменения в файл профиля на диске
RELOAD INTO KERNEL	Немедленно делает сохранённое изменение действующим, не дожидаясь следующей загрузки или перезапуска службы

WHAT	PROFILES → нажмите на имя профиля → EDITOR его открывает → внесите изменение → CHECK SYNTAX → SAVE → RELOAD INTO KERNEL.
WHY	Это полный цикл ограничения в одну строку: наблюдайте в режиме complain (VIOLATIONS), исправьте то, чего действительно не хватает (EDITOR), и только потом переключайте этот профиль в enforce (PROFILES).
RESULT	Профиль ведёт себя именно так, как отредактирован, вживую, без перезагрузки.
RISK	Неправильное редактирование профиля может позже заблокировать что-то законное в режиме enforce — CHECK SYNTAX ловит неправильно сформированные правила перед SAVE, но не логические ошибки. Если отредактированный вами профиль начинает вызывать проблемы в режиме enforce, верните его в complain из PROFILES и снова проверьте VIOLATIONS.

▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.

6.6 Advanced

ЭКВИВАЛЕНТЫ КОМАНДНОЙ СТРОКИ

```
$ sudo aa-status
# те же подсчёты, что и в заголовке вкладки PROFILES, из терминала
$ sudo aa-enforce /etc/apparmor.d/PROFILE_FILE
$ sudo aa-complain /etc/apparmor.d/PROFILE_FILE
$ sudo aa-disable /etc/apparmor.d/PROFILE_FILE
# эквивалент RELOAD INTO KERNEL после ручного редактирования
$ sudo apparmor_parser -r /etc/apparmor.d/PROFILE_FILE
```

ЧТЕНИЕ НЕОБРАБОТАННОГО ЖУРНАЛА АУДИТА

```
$ sudo grep apparmor /var/log/audit/audit.log | tail -20
# или, если не запущен выделенный демон аудита:
$ sudo dmesg | grep -i apparmor
```

Полезные поля при чтении строки вручную: `apparmor="DENIED"` (enforce) или `apparmor="ALLOWED"` (complain, только зарегистрировано), `operation=` (что было предпринято), `profile=` (какой профиль принял решение — соответствует тому, что открывает EDITOR), `name=` (путь или ресурс), `comm=` (процесс).

ПОЧЕМУ МАССОВЫЙ ALL→ENFORCE РИСКОВАН

Переводить профиль в enforce индивидуально после того, как его журнал VIOLATIONS какое-то время оставался тихим — это безопасный паттерн: вы знаете, профиль за профилем, что обычное использование его не активирует. ALL→ENFORCE пропускает этот период наблюдения для всех профилей сразу, включая те, которые вы, возможно, только что установили из COMMUNITY и никогда по-настоящему не использовали. Профиль, применённый прежде, чем его прожили, может заблокировать законную функцию именно в момент, когда вы используете её впервые.

Предпочитайте цикл по одному; оставьте массовые действия для ALL→COMPLAIN (всегда безопасно — только ослабляет) или ALL→DISABLE при устранении неполадок.

ФАЙЛЫ, КОТОРЫЕ ЗАТРАГИВАЕТ ЭТА ПАНЕЛЬ

ПУТЬ	РОЛЬ
`/etc/apparmor.d/`	Один файл профиля на каждую ограниченную программу — то, что перечисляет PROFILES и открывает EDITOR
`/etc/default/apparmor`	Режим по умолчанию при загрузке (`APPARMOR=complain`) и состояние enable/disable
`/etc/default/grub`	Параметры ядра `apparmor=1 security=apparmor`, которые проверяет STATUS

CHAPTER 7

Privacy

Два независимых инструмента, каждый со своим собственным процессом из трёх шагов install → enable → running, и мост между ними, как только оба работают.

7.1 Визуальное руководство — SearXNG, шаг за шагом

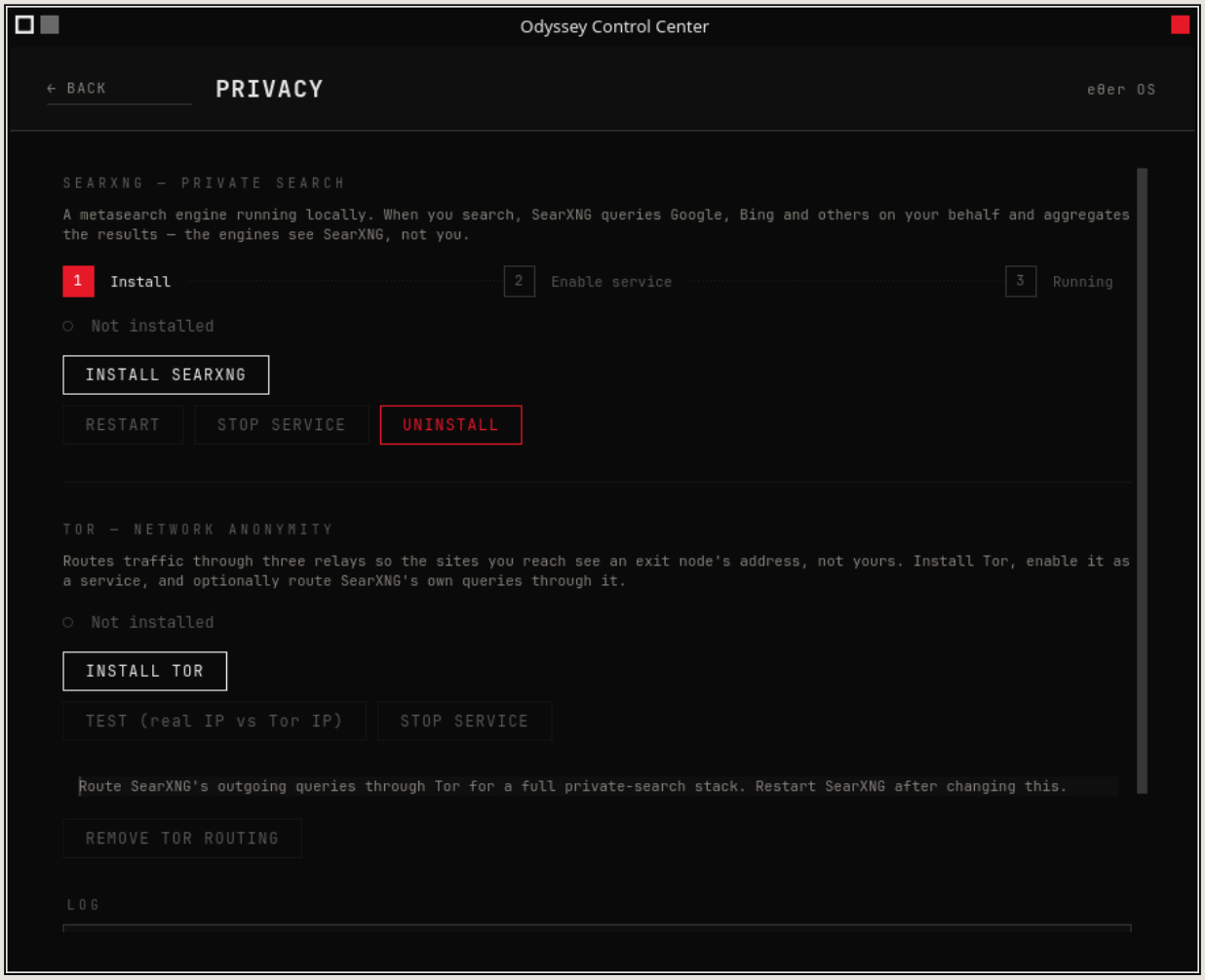

FIG · Панель в состоянии покоя: индикатор SearXNG на шаге 1 «Install», статус «Not installed», одна кнопка INSTALL SEARXNG; индикатор Tor также показывает «Not installed» с INSTALL TOR ниже.

SearXNG — это метапоисковая система, работающая локально на вашей собственной машине — когда вы ищете, именно SearXNG опрашивает Google, Bing и другие от вашего имени, так что эти поисковики видят запрос от SearXNG, а не напрямую от вас.

1 **INSTALL SEARXNG**

Нажмите. Журнал показывает реальную установку в процессе — клонирование исходного кода, создание системного пользователя, настройку виртуального окружения Python.

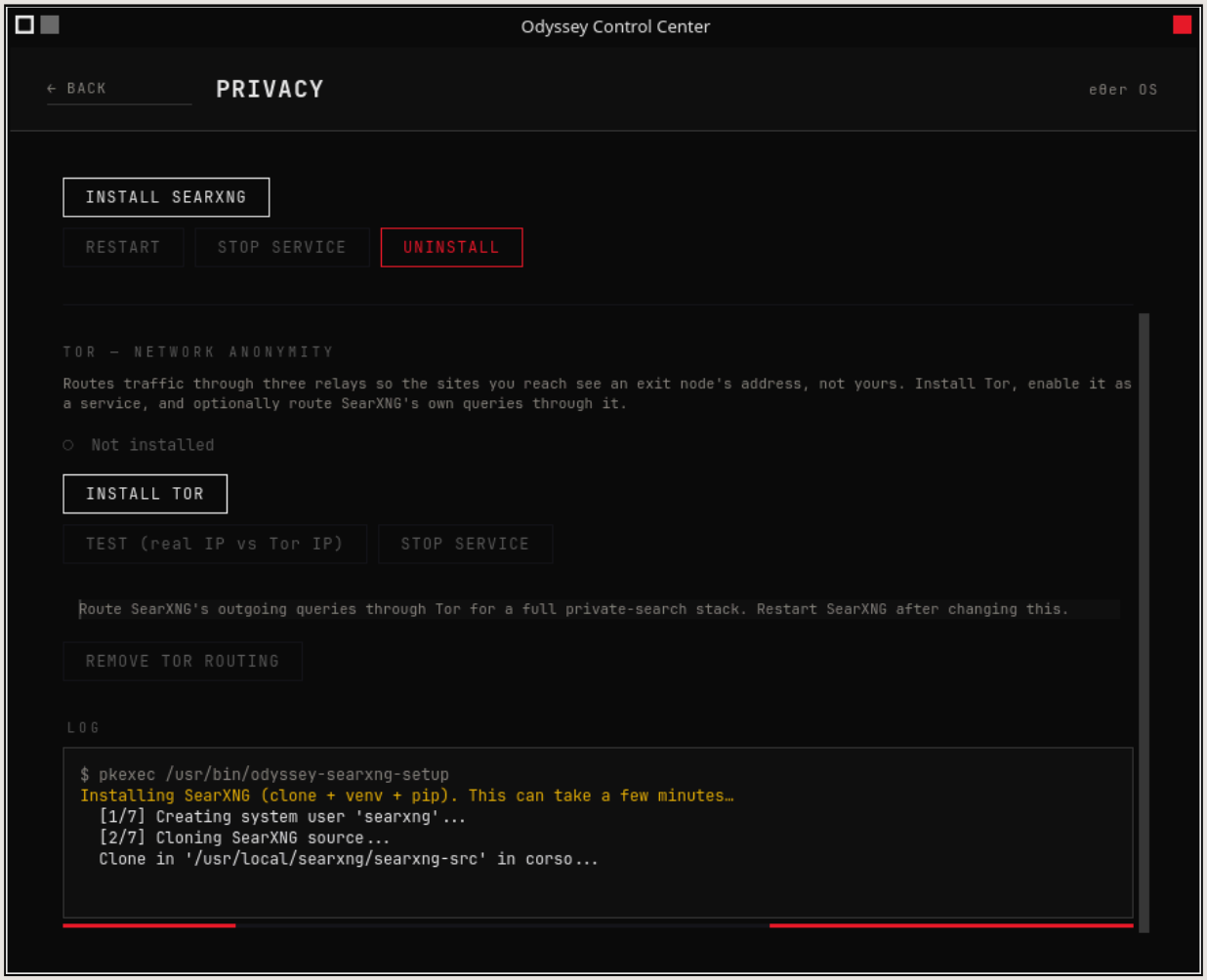

FIG · В середине установки: журнал передаёт реальный вывод — «Installing SearXNG (clone + venv + pip). This can take a few minutes…» с видимыми шагами 1/7 и 2/7.

2 ENABLE SERVICE

После установки индикатор переходит на шаг 2, и строка статуса становится жёлтой: «Installed — service not enabled». Теперь одна кнопка: ENABLE SERVICE.

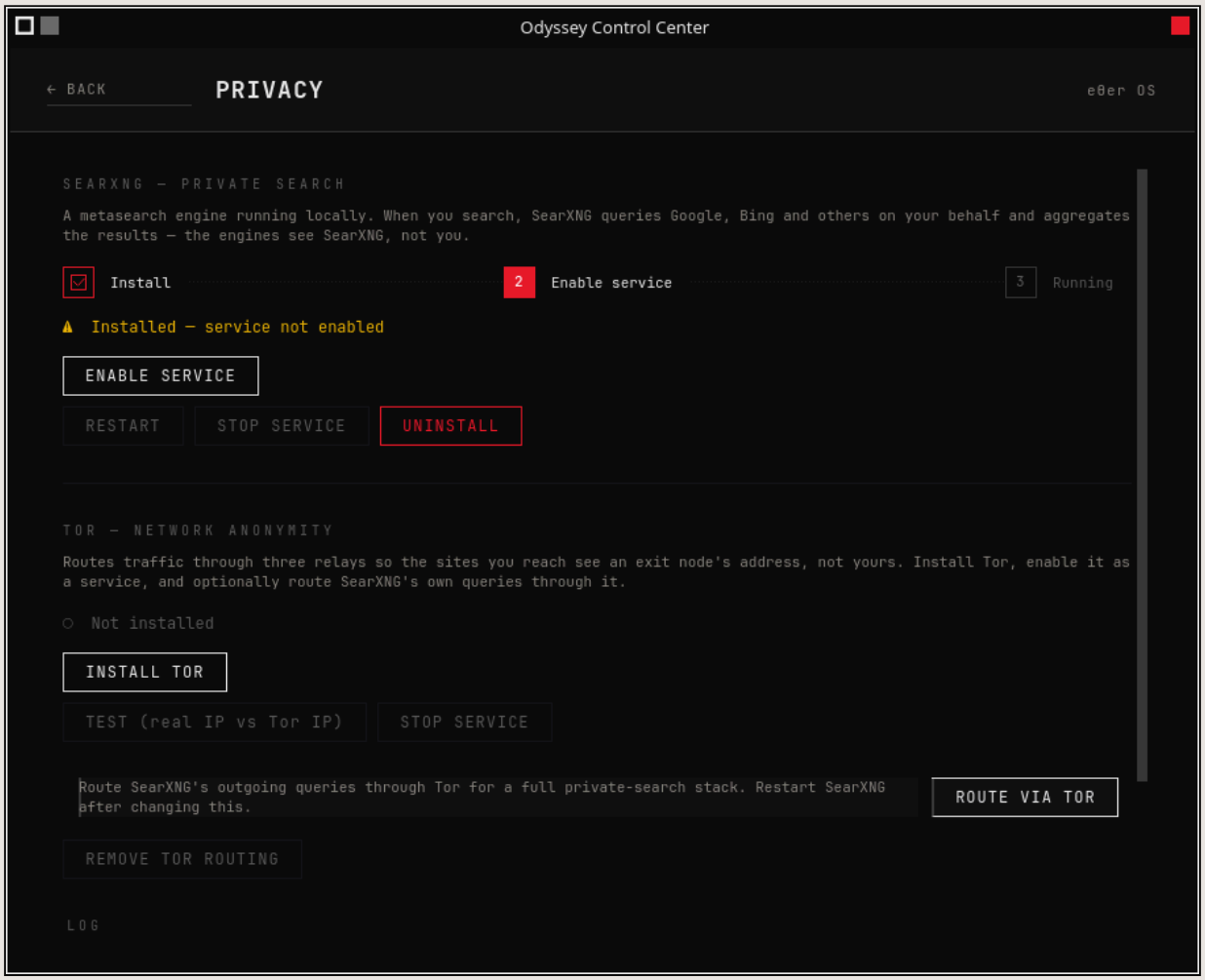

FIG · Активен шаг 2: «Installed — service not enabled», ENABLE SERVICE как единственная активная кнопка, RESTART/STOP SERVICE/UNINSTALL всё ещё отключены ниже.

3 Running

После включения индикатор достигает шага 3, и вся строка управления становится активной. Как только сервис работает, **OPEN IN BROWSER** ведёт вас прямо туда — SearXNG по умолчанию слушает на `http://localhost:8888`.

WHAT	Privacy → INSTALL SEARXNG → ENABLE SERVICE → OPEN IN BROWSER.
WHY	Каждая коммерческая поисковая система со временем строит профиль на основе ваших запросов; локальный экземпляр метапоиска означает, что только вы когда-либо видите неагрегированный запрос.
RESULT	Приватная поисковая система, работающая на вашей собственной машине, доступная в вашем браузере, опрашивающая крупные поисковики от вашего имени, не подвергая вас прямому контакту с ними.
RISK	Низкий — это локальный веб-сервис, как и любой другой. RESTART, если он ведёт себя неправильно, STOP SERVICE, чтобы приостановить его без удаления, UNINSTALL, чтобы удалить его полностью.

7.2 Визуальное руководство — Tor, шаг за шагом

Та же форма из трёх шагов, независимый индикатор, прямо под разделом SearXNG.

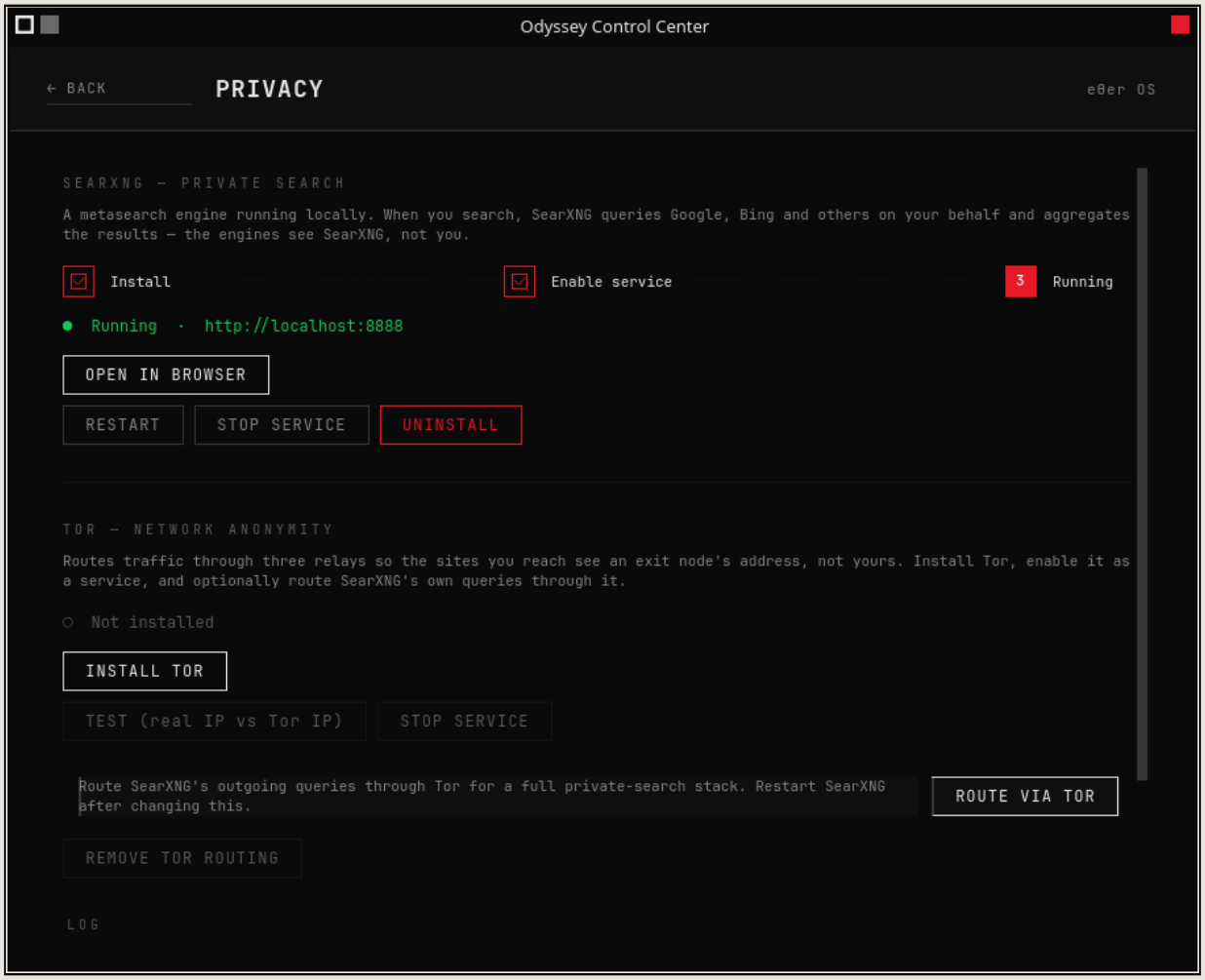

FIG · SearXNG полностью работает выше; раздел Tor ниже всё ещё в «Not installed» с INSTALL TOR как единственной активной кнопкой.

Tor направляет ваш трафик через три релея, прежде чем он достигнет пункта назначения, так что сайт, который вы посещаете, видит адрес выходного релея, а не ваш.

1 **INSTALL TOR**

Журнал подтверждает установку пакета и показывает сервис, связанный в runit.

2 **ENABLE SERVICE**

Статус меняется на «Installed — service not enabled» — тот же паттерн, что и у SearXNG.

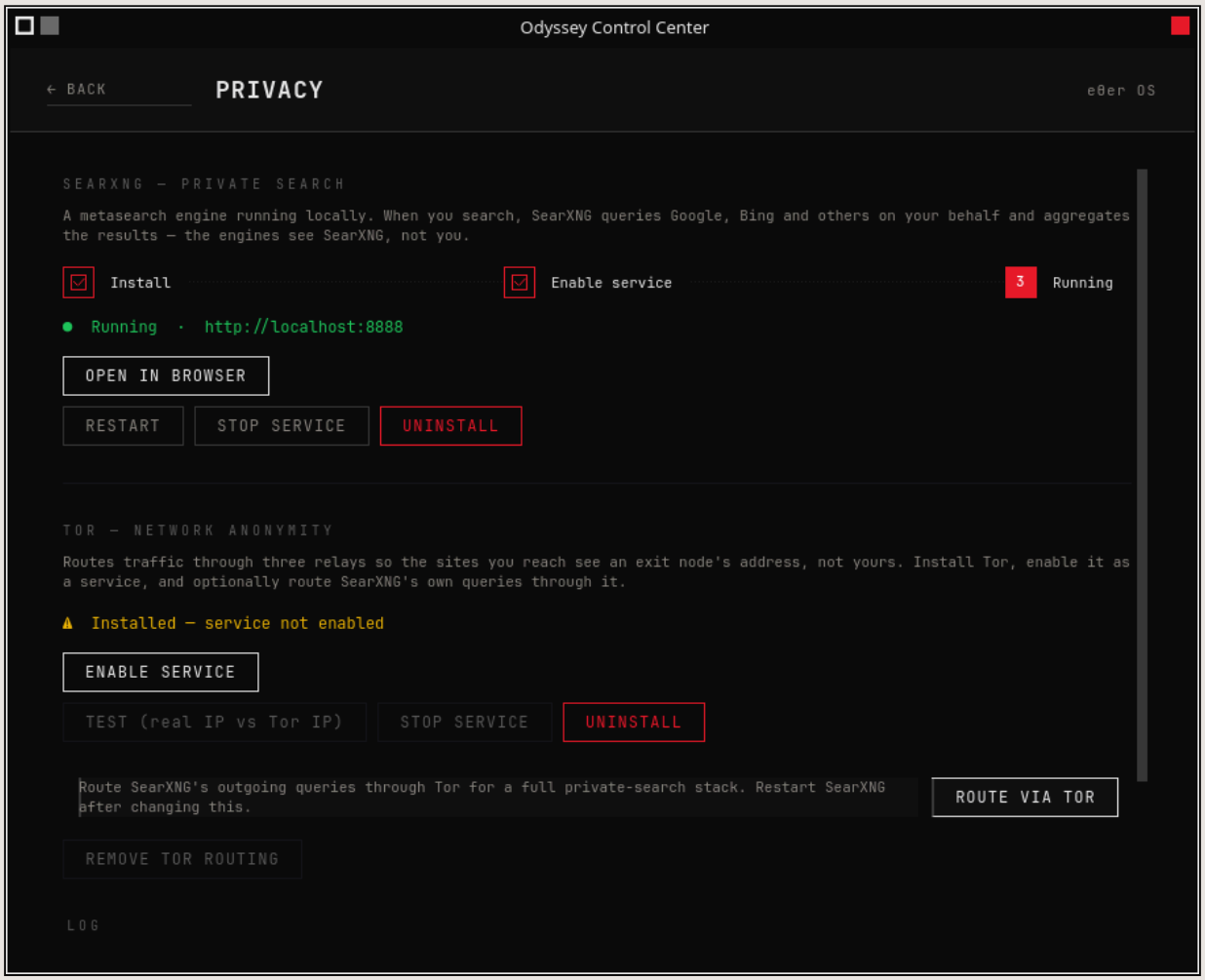

FIG · Tor на шаге 2: «Installed — service not enabled», ENABLE SERVICE активна, TEST и STOP SERVICE всё ещё отключены.

3 Running, и проверьте это

После включения появляется выделенная кнопка **TEST (real IP vs Tor IP)** — используйте её. Она сравнивает ваш реальный адрес с тем, который представляет Tor, бок о бок, чтобы у вас было реальное доказательство того, что это работает, а не просто индикатор состояния, которому приходится просто верить.

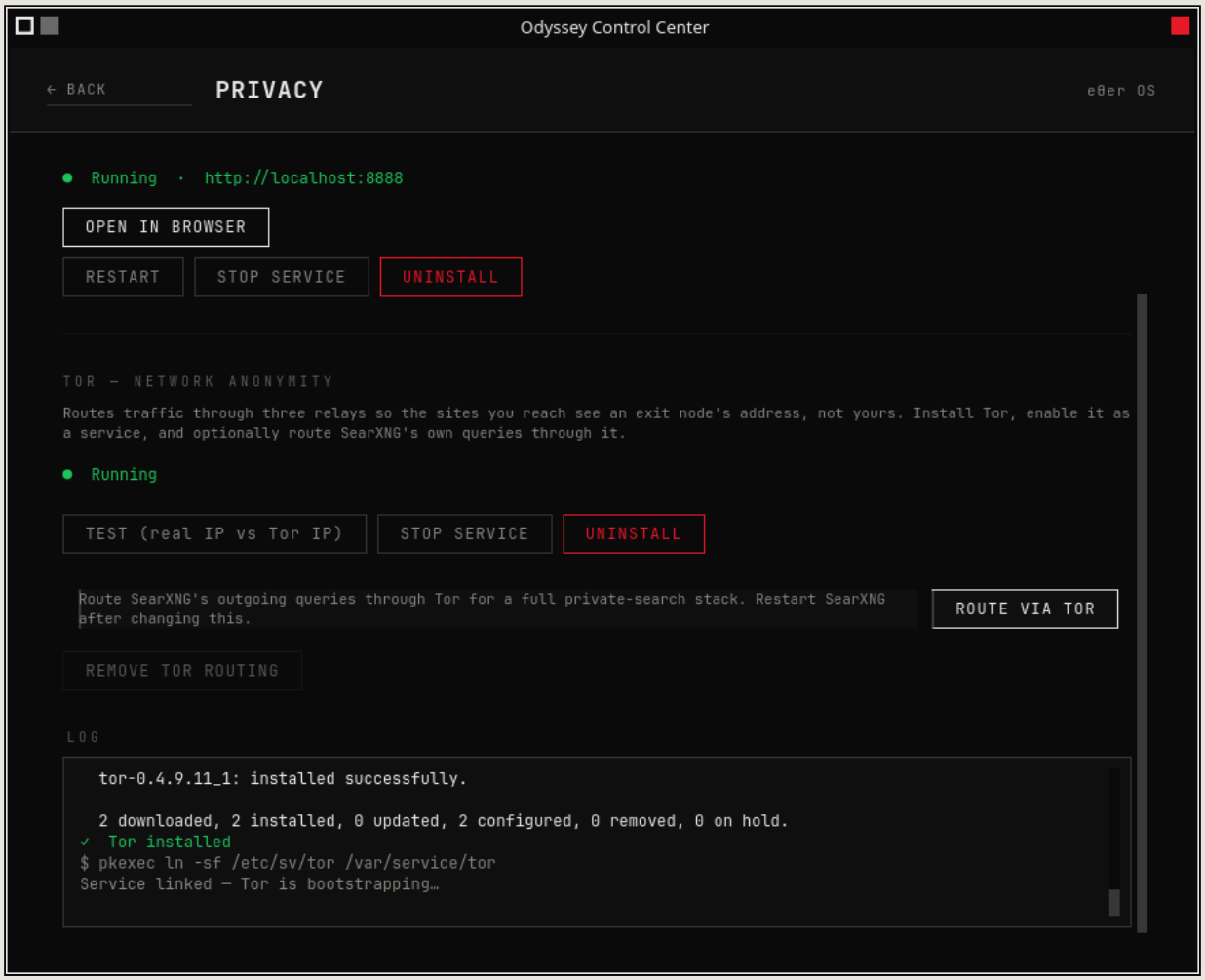

FIG · Оба раздела зелёные и работают: SearXNG на localhost:8888 с OPEN IN BROWSER, Tor работает с TEST / STOP SERVICE / UNINSTALL — все активны.

WHAT	Privacy → INSTALL TOR → ENABLE SERVICE → TEST (real IP vs Tor IP) для подтверждения.
WHY	Направление конфиденциального просмотра через Tor скрывает ваш адрес от посещаемых сайтов — TEST даёт вам доказательство, а не просто утверждение.
RESULT	Работающий сервис Tor, через который могут направляться другие приложения; TEST показывает два действительно разных адреса, если всё работает.
RISK	Трафик через Tor заметно медленнее — это цена за анонимность. STOP SERVICE приостанавливает его; UNINSTALL удаляет.

7.3 Визуальное руководство — объединяя их

Как только оба работают, появляется встроенное уведомление, предлагающее направить через Tor и собственные запросы SearXNG — не только ваши запросы к SearXNG, но и запросы SearXNG к Google и Bing от вашего имени.

WHAT	Privacy → уведомление «Route SearXNG's outgoing queries through Tor» → ROUTE VIA TOR → RESTART в разделе SearXNG, чтобы это вступило в силу.
WHY	Без этого сам SearXNG продолжает напрямую опрашивать крупные поисковые системы — приватно с вашей стороны, видимо с их стороны. Направление собственного трафика SearXNG через Tor закрывает этот пробел для полностью приватного поискового стека.
RESULT	Исходящие запросы SearXNG, а не только ваши запросы к нему, проходят через Tor.
RISK	Результаты загружаются заметно медленнее, по той же причине, что и любой трафик, направленный через Tor. REMOVE TOR ROUTING отменяет это, снова с последующим перезапуском.

▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.

7.4 Advanced

ЧТО НА САМОМ ДЕЛЕ ВЫПОЛНЯЕТ INSTALL SEARXNG

```
$ pkexec /usr/bin/odyssey-searxng-setup
# создаёт выделенного системного пользователя 'searxng',
# клонирует исходный код в /usr/local/searxng/searxng-src,
# создаёт venv Python и устанавливает зависимости через pip
```

СЕРВИСЫ И ПОРТЫ, ВРУЧНУЮ

```
# собственный сервис runit у SearXNG
$ sudo sv status searxng
$ sudo sv restart searxng
# сервис runit у Tor — шаг включения панели буквально это
$ sudo ln -sf /etc/sv/tor /var/service/tor
$ sudo sv status tor
# проверьте, что Tor действительно направляет трафик
$ curl -socks5 localhost:9050 https://check.torproject.org/api/ip
```

НАПРАВЛЕНИЕ SEARXNG ЧЕРЕЗ TOR ВРУЧНУЮ

SOCKS-прокси Tor по умолчанию слушает на `127.0.0.1:9050`. Направить SearXNG через него означает указать его исходящему HTTP-клиенту на этот прокси — настройка находится в собственном файле `settings.yml` SearXNG, в разделе конфигурации исходящих запросов, а не в общесистемной переменной прокси. Кнопка ROUTE VIA TOR панели напрямую редактирует этот файл и перезапускает сервис; сделать это вручную означает найти и отредактировать тот же блок `outgoing:` в `/usr/local/searxng/searxng-src/searx/settings.yml`.

ПОЧЕМУ SEARXNG НУЖЕН ПЕРЕЗАПУСК ПОСЛЕ ИЗМЕНЕНИЙ МАРШРУТИЗАЦИИ

SearXNG читает свою конфигурацию прокси только один раз, при запуске — это не настройка, перезагружаемая на лету. Это верно независимо от того, меняете ли вы её через панель или вручную, поэтому оба пути заканчиваются одним и тем же шагом RESTART.

ФАЙЛЫ, КОТОРЫЕ ЗАТРАГИВАЕТ ЭТА ПАНЕЛЬ

ПУТЬ	РОЛЬ
`/usr/local/searxng/searxng-src/`	Собственная установка SearXNG — исходный код, venv, конфигурация
`/etc/sv/searxng`, `/etc/sv/tor`	Определения сервисов runit
`/var/service/`	Где включение сервиса связывает его с работающим деревом супервизии

CHAPTER 8

Containers

Управление контейнерами от начала до конца: настройка движка, список работающих контейнеров, отобранный каталог для старта, необработанные команды для всего, чего в нём нет, и использование диска — всё в одной прокручиваемой панели.

8.1 Визуальное руководство — движок

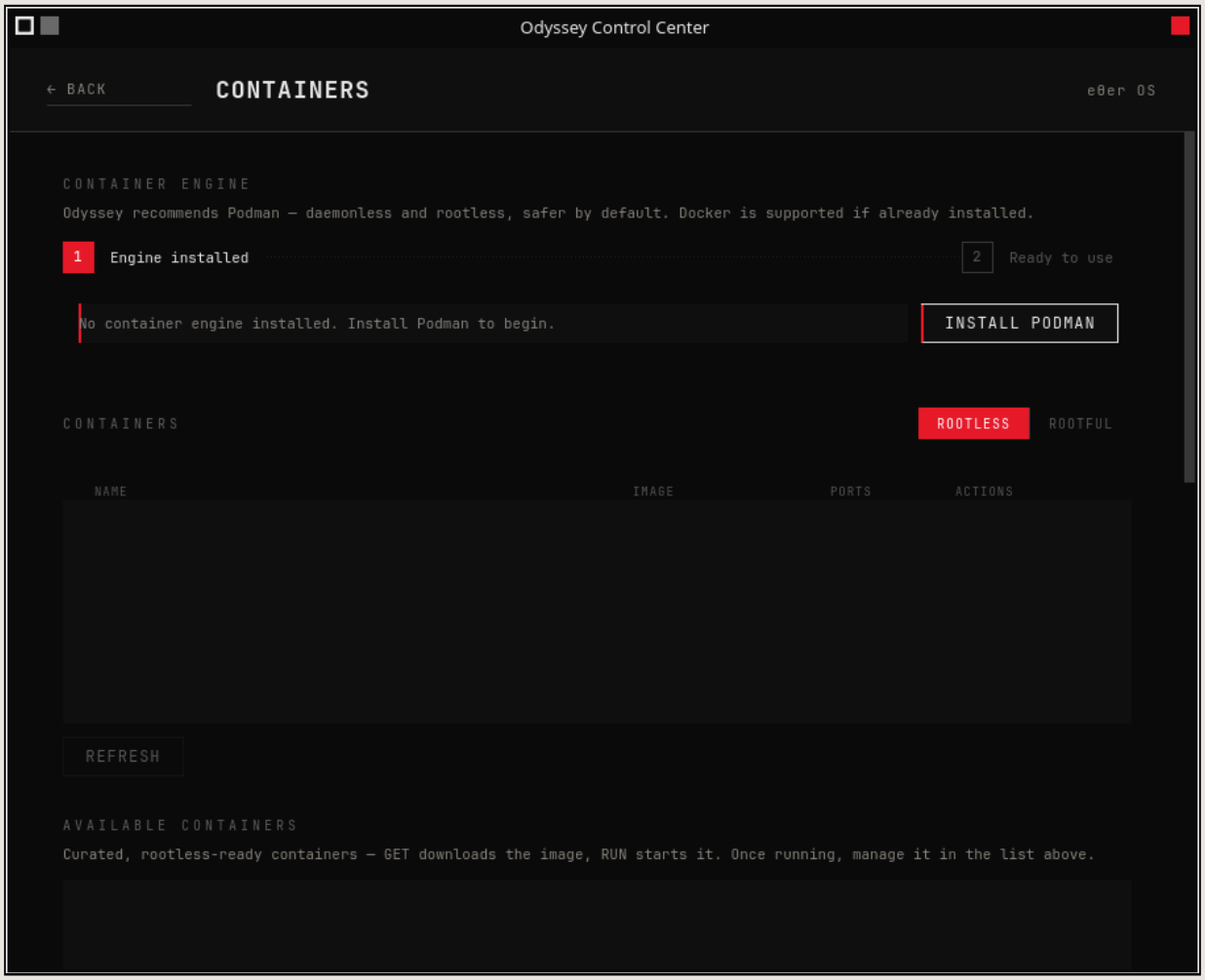

FIG · Шаг 1 из 2: «No container engine installed. Install Podman to begin», с INSTALL PODMAN как единственным активным действием. Разделы Containers list и Available containers ниже видны, но в пустом состоянии.

Odyssey рекомендует **Podman** — без демона и rootless, что означает, что контейнеры работают без постоянно запущенной фоновой службы и, по умолчанию, без прав root. **Docker** также поддерживается, если это уже то, чем вы пользуетесь.

1 **INSTALL PODMAN**

Журнал показывает реальную установку пакета.

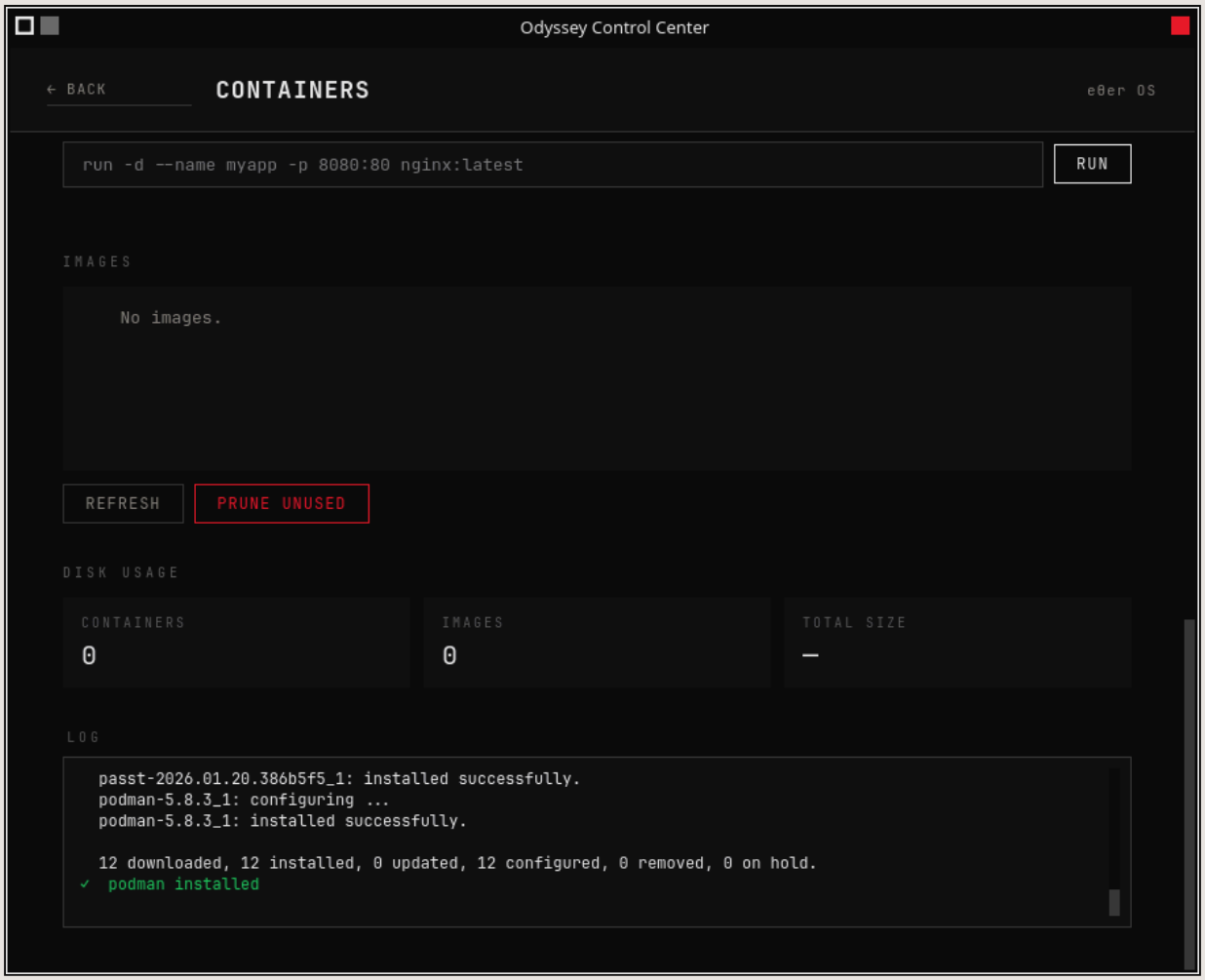

FIG · После установки: «Engine installed» отмечено красным, шаг 2 «Ready to use», информационная строка гласит «podman version 5.8.3 · Podman installed · Docker not installed», и кнопка INSTALL DOCKER TOO для тех, кто тоже хочет Docker параллельно.

2 Ready to use

Движок активен. **INSTALL DOCKER TOO** остаётся доступной, если вы хотите оба движка бок о бок — они сосуществуют без конфликта.

8.2 Визуальное руководство — rootless vs. rootful, и пустое состояние

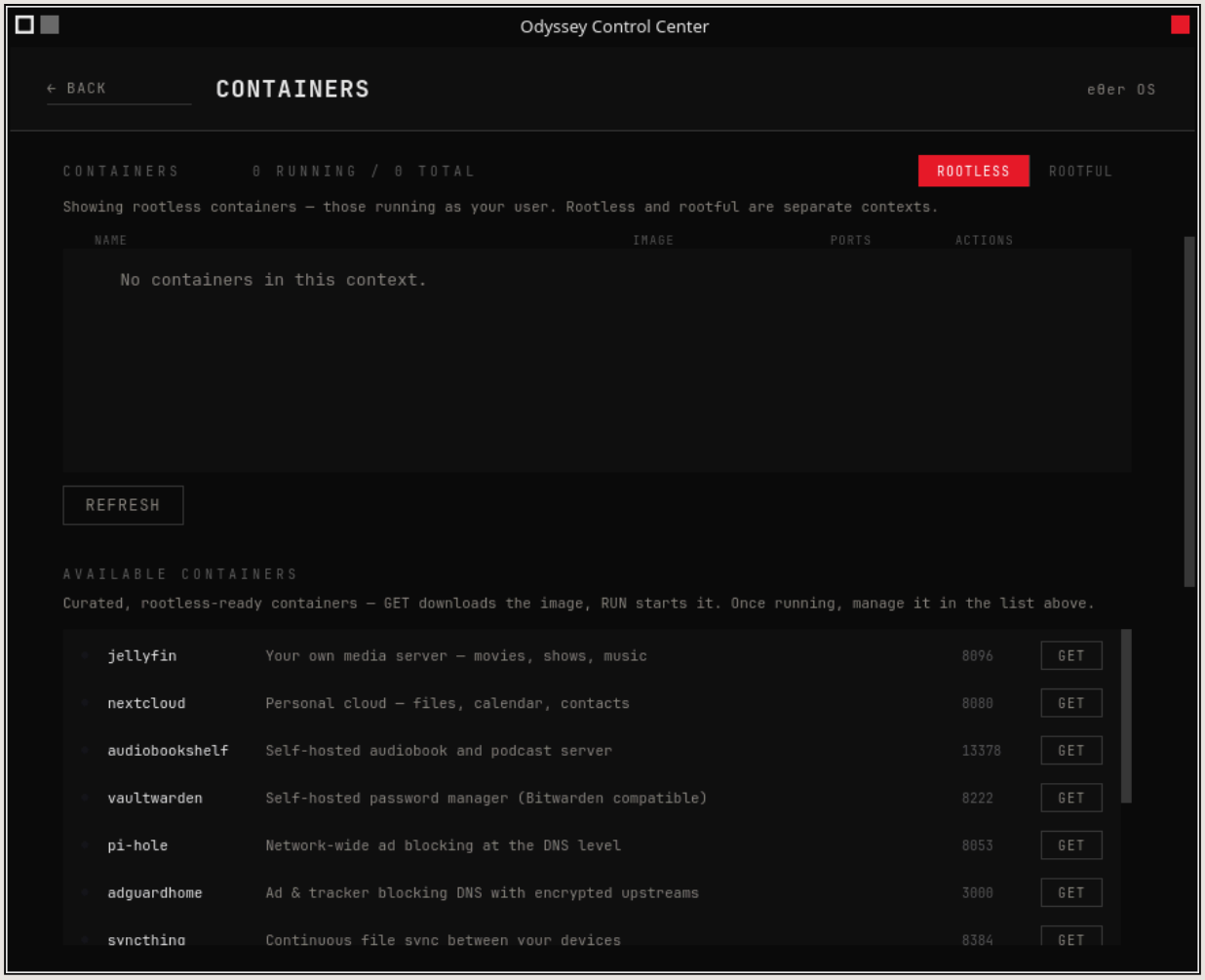

FIG · ROOTLESS выбран (выделен красным) рядом с ROOTFUL, «0 RUNNING / 0 TOTAL», и «No containers in this context.» под заголовками столбцов NAME / IMAGE / PORTS / ACTIONS.

Переключатель **ROOTLESS / ROOTFUL** вверху меняет, какой контекст вы просматриваете и которым управляете — это два полностью раздельных пространства имён контейнеров, а не фильтр на объединённом списке. Odyssey показывает ROOTLESS по умолчанию, и этот выбор сделан намеренно:

> **КАКОЙ ИЗ ДВУХ ВАМ СЛЕДУЕТ ИСПОЛЬЗОВАТЬ**
>
> Rootless — более безопасный выбор, и именно к нему направляет вас панель: контейнер, вырвавшийся из ограничения в режиме rootless, всё равно не может сделать ничего, чего не могла бы уже сделать ваша собственная учётная запись пользователя. Rootful существует для конкретных случаев, которым он нужен — определённые сетевые порты с низким номером, некоторый доступ к оборудованию — и должен быть исключением, а не привычкой.

8.3 Визуальное руководство — отобранный каталог, от начала до конца

Прокрутите до **Available containers** — уже готовый список, чтобы панель никогда не была пустой коробкой в ожидании того, что вы уже знаете названия образов наизусть. Каждая строка показывает контейнер, описание в одну строку и порт, который он использует.

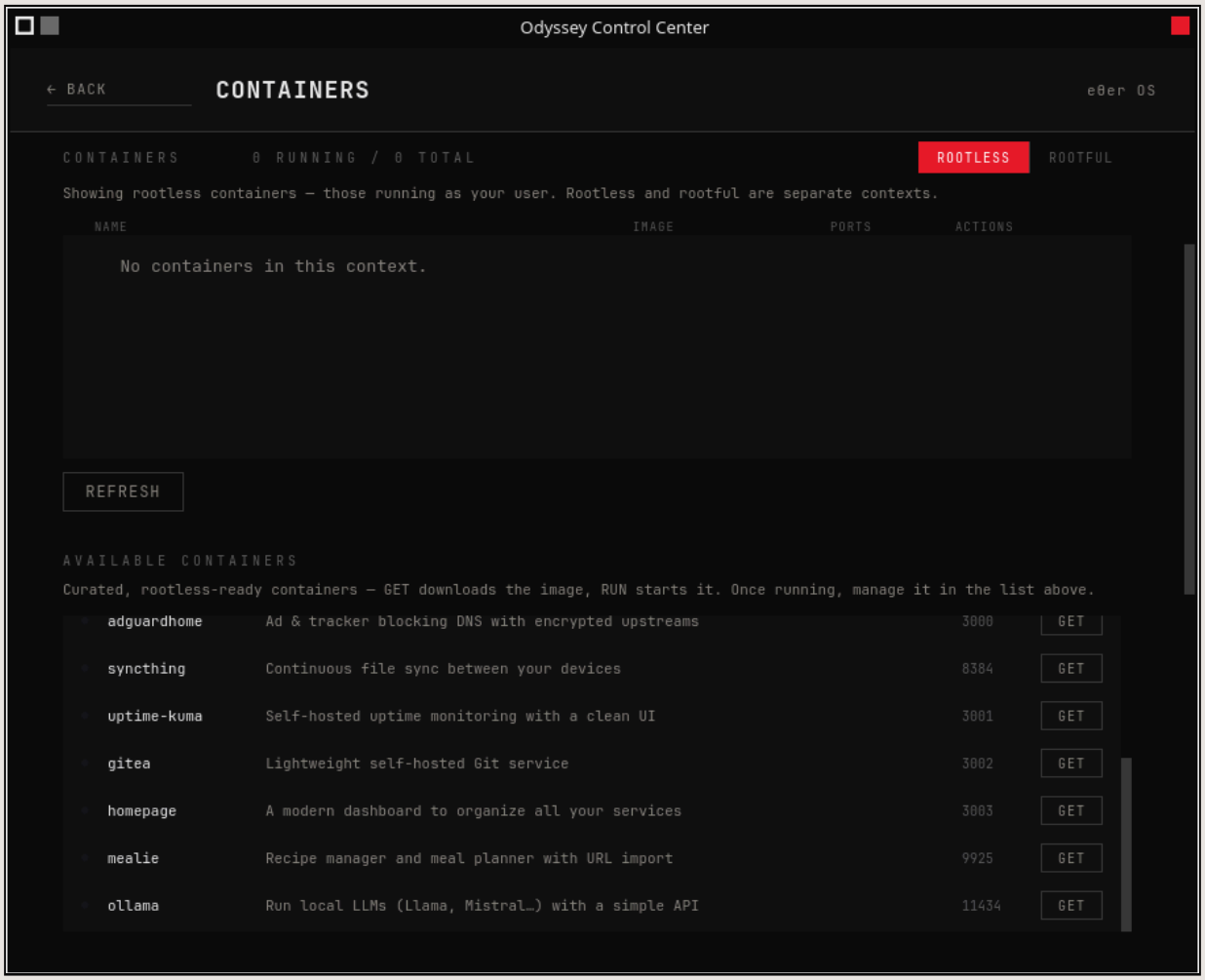

FIG · Список каталога: adguardhome (порт 3000), syncthing (8384), uptime-kuma (3001), gitea (3002), homepage (3003) — с кнопкой GET в каждой строке.

Этот номер порта — тот, на котором вы получите доступ к сервису после запуска — `homepage` с `3003`, например, означает `http://localhost:3003` после активации.

1 **GET**

Нажмите GET на `homepage`. Это только скачивает образ контейнера — ничего ещё не работает.

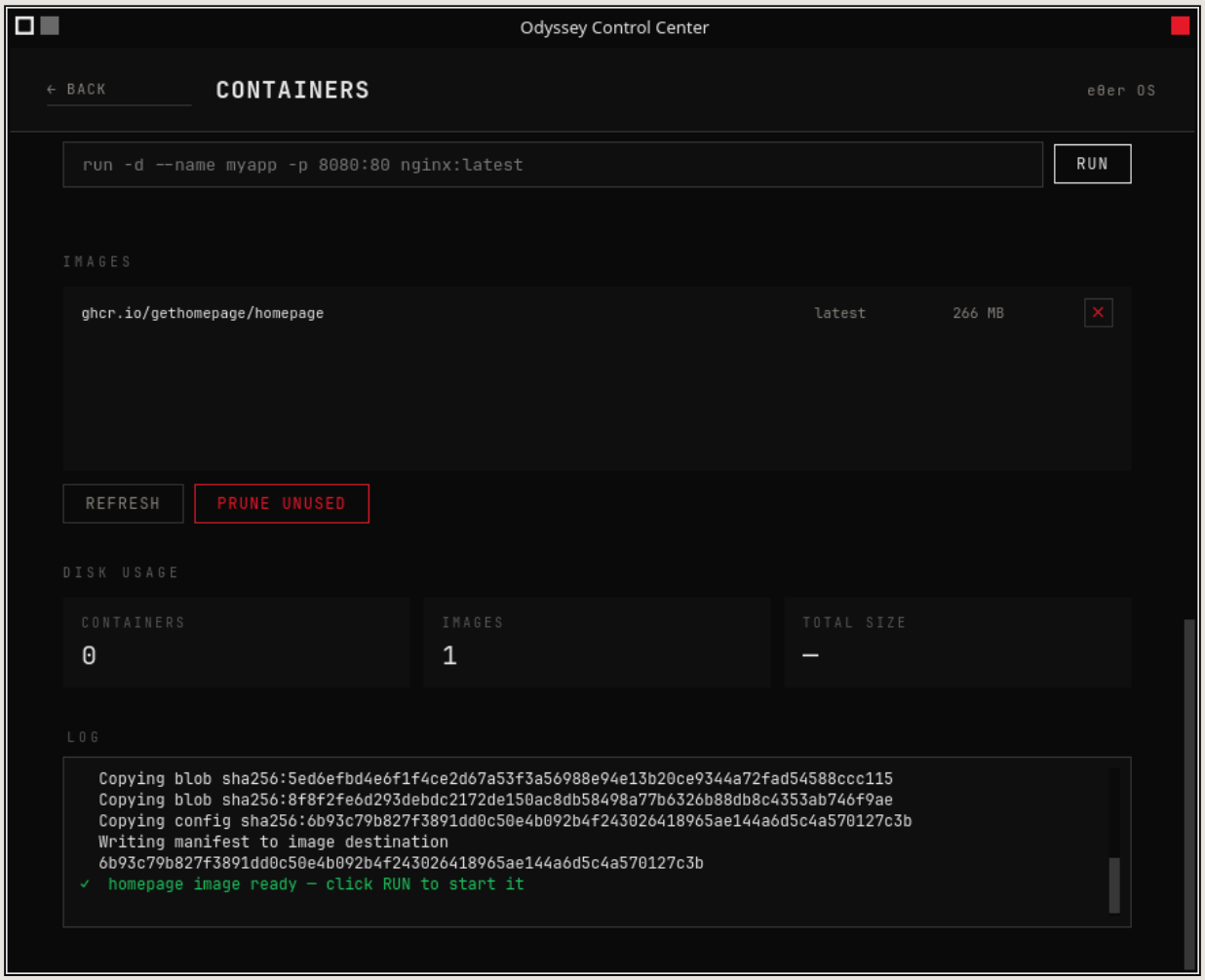

FIG · Журнал передаёт реальную загрузку образа: «Copying blob sha256:5ed6…», «Writing manifest to image destination», заканчиваясь зелёным — «homepage image ready — click RUN to start it».

2 RUN

Строка кнопки GET теперь показывает RUN вместо неё — нажмите, чтобы действительно запустить контейнер.

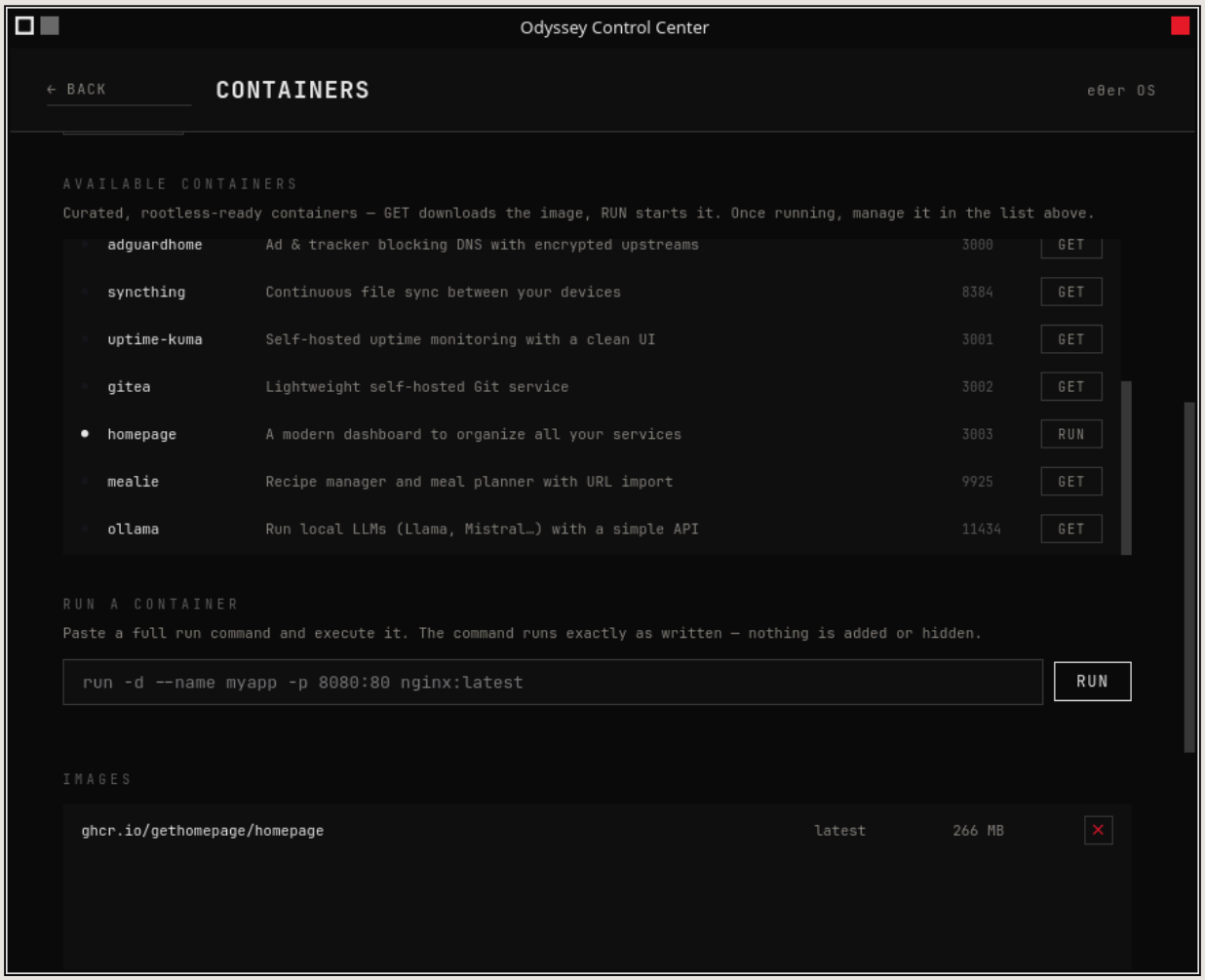

FIG · homepage теперь указан в разделе CONTAINERS выше, зелёная точка, путь образа ghcr.io/gethomepage/homepage, порты «0.0.0.0:3003→…», со своей собственной строкой значков действий; всё ещё виден ниже в Available Containers с новой кнопкой RUN для запуска ещё одного экземпляра.

После запуска `homepage` появляется в двух местах одновременно: как активная запись в списке **Containers** вверху (с портом и значками действий), и всё ещё в каталоге ниже на случай, если вы захотите запустить второй экземпляр позже.

WHAT	Containers → Available containers → GET на чём-то из каталога → RUN после загрузки.
WHY	Разделение на два шага (скачать, затем запустить) означает, что вы можете получить образ заранее, не обязуясь запускать его сразу — полезно на медленном соединении или когда вы просто просматриваете, что доступно.
RESULT	Работающий контейнер, видимый в основном списке с привязкой портов, управляемый оттуда далее.
RISK	Низкий — самостоятельно размещённые сервисы отобранного каталога выбраны так, чтобы быть дружественными к rootless и низкорисковыми по умолчанию. Уязвимость такая же, как при запуске любого веб-сервиса: любой порт, на котором он слушает, доступен согласно вашим настройкам Firewall.

8.4 Визуальное руководство — список контейнеров и его действия

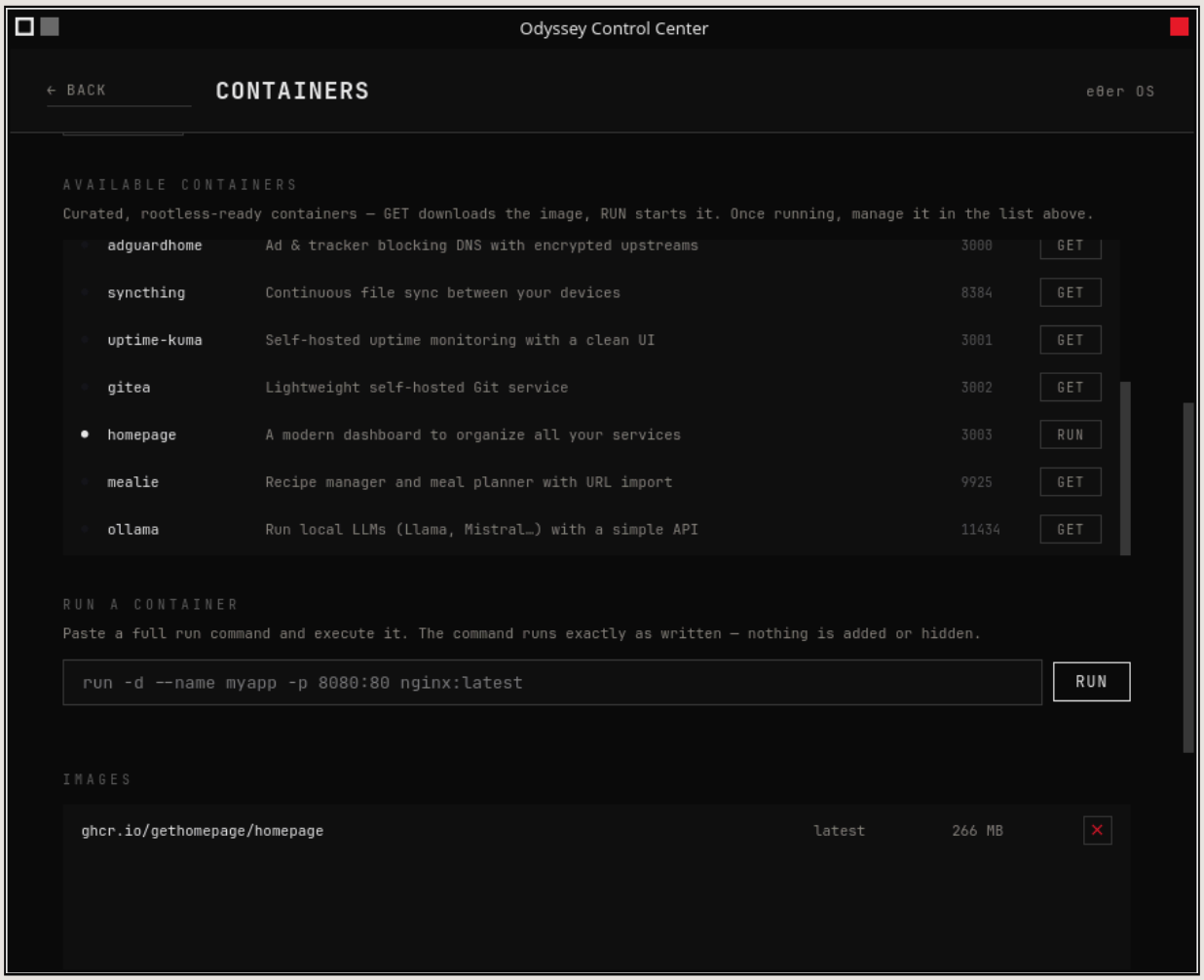

FIG · Строка homepage в списке Containers, с её пятью значками действий, видимыми справа — теми, что объясняются ниже.

Каждая строка работающего контейнера заканчивается набором значков — наведите на любой, чтобы увидеть его подпись, но вот что делает каждый:

ЗНАЧОК	ДЕЙСТВИЕ
открыть (стрелка)	Открывает сервис напрямую в вашем браузере — быстрый путь, как только что-то работает
остановить / запустить (квадрат / треугольник)	Останавливает контейнер, не удаляя его, или снова запускает
перезапустить (круговая стрелка)	Останавливает и запускает за одно действие — решение для случая «работает, но не отвечает»
логи (значок строк)	Открывает собственный вывод журнала контейнера — первое место, куда стоит заглянуть, если что-то не работает
удалить (×)	Полностью удаляет контейнер — сам образ остаётся скачанным, отдельно перечисленным в Images

8.5 Визуальное руководство — запуск контейнера и его открытие

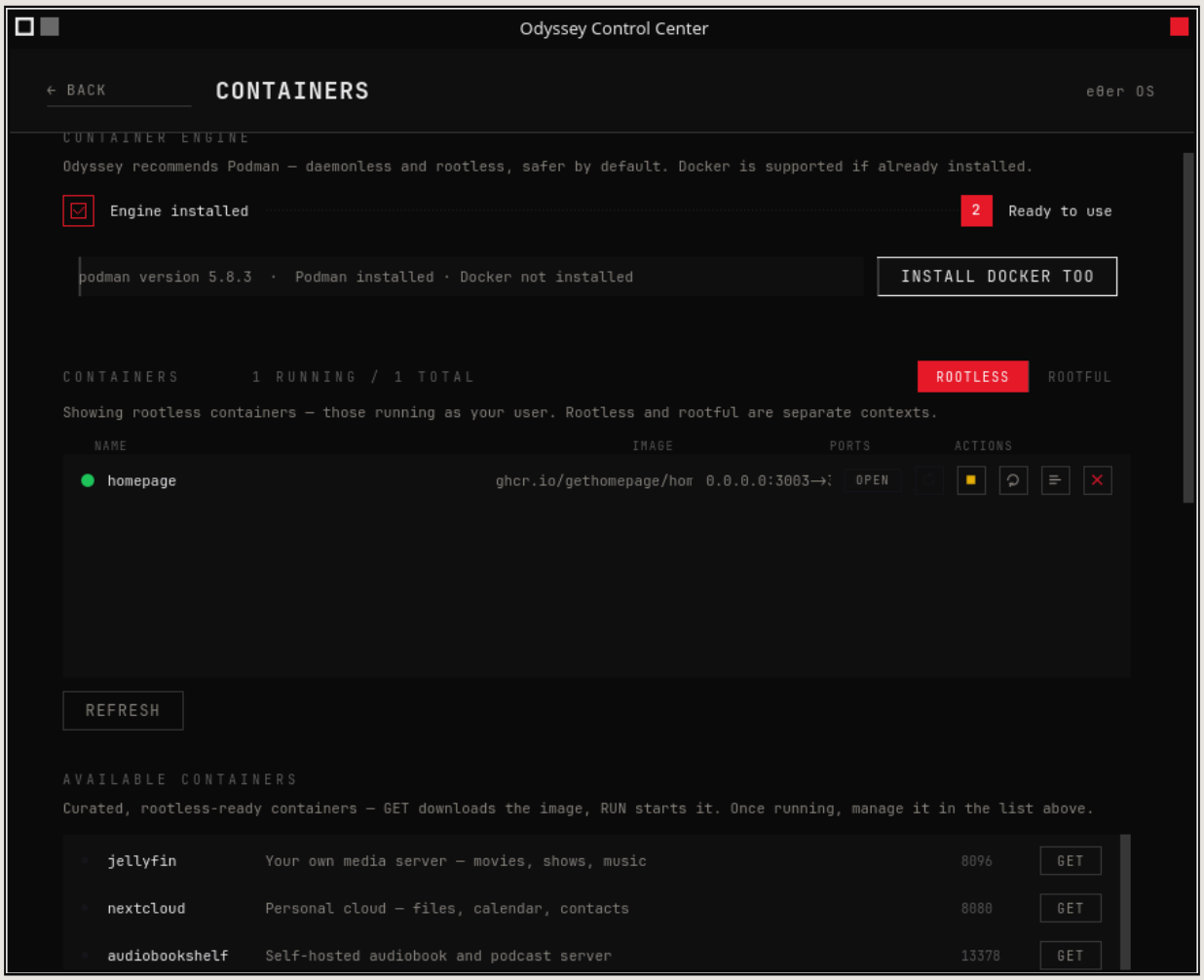

FIG · Поле «Run a container» с примером команды в качестве текста- подсказки, кнопка RUN, и раздел Images ниже, перечисляющий ghcr.io/gethomepage/homepage · latest · 266 MB с кнопкой удаления.

Для всего, чего нет в каталоге, **Run a container** принимает полную команду `podman run` (или `docker run`) и выполняет её точно так, как она написана — ничего не добавляется, ничего не скрывается.

WHAT	Containers → «Run a container» → вставьте полную команду run (например, `run -d --name myapp -p 8080:80 nginx:latest`) → RUN.
WHY	Отобранный каталог покрывает распространённые случаи; это выход для конкретного образа, конкретных флагов, конкретных портов, которые каталог не предлагает.
RESULT	Контейнер запускается точно так, как указывает команда, и присоединяется к основному списку Containers после запуска.
RISK	То, что делает сама команда — это точная передача, а не песочница. Относитесь к ней с той же осторожностью, с какой вы бы ввели ту же команду в терминале.

Ниже **Images** перечисляет каждый скачанный образ независимо от того, работает ли сейчас от него контейнер, с **REFRESH** и **PRUNE UNUSED** для освобождения места от образов, на которые никто больше не ссылается.

ОТКРЫТИЕ РАБОТАЮЩЕГО КОНТЕЙНЕРА

Нажатие OPEN на строке работающего контейнера — то же действие открытия из строки значков выше — запускает ваш браузер по умолчанию напрямую по его адресу и порту, например `http://localhost:3003` для `homepage`. Не нужно самостоятельно запоминать номер порта, как только он работает.

8.6 Использование диска

В самом низу панели — три показателя в реальном времени: контейнеры, образы, общий размер — чтобы вы всегда знали, во сколько вам обходятся контейнеры на диске, не открывая терминал.

```
▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.
```

8.7 Advanced

ЭКВИВАЛЕНТЫ КОМАНДНОЙ СТРОКИ

```
# то, что панель называет «GET»
$ podman pull ghcr.io/gethomepage/homepage
# то, что панель называет «RUN» на записи каталога
$ podman run -d -name homepage -p 3003:3000 ghcr.io/gethomepage/homepage
# перечисляет работающие контейнеры — контекст rootless
$ podman ps
# то же самое, rootful
$ sudo podman ps
# логи, остановка, перезапуск, удаление — строка значков, вручную
$ podman logs -f homepage
$ podman stop homepage
$ podman restart homepage
$ podman rm homepage
# образы и использование диска
$ podman images
$ podman system df
$ podman image prune
```

ROOTLESS VS. ROOTFUL, ПОД КАПОТОМ

Rootless Podman запускает контейнеры от имени вашего собственного пользователя, внутри пространства имён пользователя — «root» контейнера сопоставляется с диапазоном UID без привилегий на хосте, поэтому даже полностью скомпрометированный процесс контейнера всё равно ограничен разрешениями вашей собственной учётной записи. Rootful Podman (`sudo podman ...`) работает традиционным способом, с root контейнера, сопоставленным с реальным root хоста. Переключатель панели на самом деле просто переключается между запуском обычного `podman` или `sudo podman` — по-настоящему раздельные каталоги состояния, раздельные списки контейнеров, ничего общего между двумя контекстами по умолчанию.

```
# где каждый контекст хранит своё собственное состояние
$ podman info -format '{{.Store.GraphRoot}}'
# rootless:  /.local/share/containers/storage
# rootful: /var/lib/containers/storage
```

ПОЧЕМУ НЕКОТОРЫМ ЗАПИСЯМ КАТАЛОГА НУЖНЫ КОНКРЕТНЫЕ ОТКРЫТЫЕ ПОРТЫ

Привязка портов контейнера (`3003:3000` в примере с homepage) по умолчанию делает сервис доступным **только с этой машины** — Podman привязывает его к `localhost`, если не указано иное. Доступ к нему с другого устройства в вашей сети дополнительно требует, чтобы этот порт был открыт в панели Firewall (Глава 4) — эти две системы являются независимыми уровнями, и работающий контейнер с привязанным портом автоматически не доступен из интернета или локальной сети только потому, что Podman на нём слушает.

CHAPTER 9

Gaming Mode

Главный переключатель вверху, строка статуса в реальном времени, и четыре вкладки: **PACKAGES**, **GAMEMODE**, **MANGOHUD**, **PROTON**.

9.1 Визуальное руководство — главный переключатель

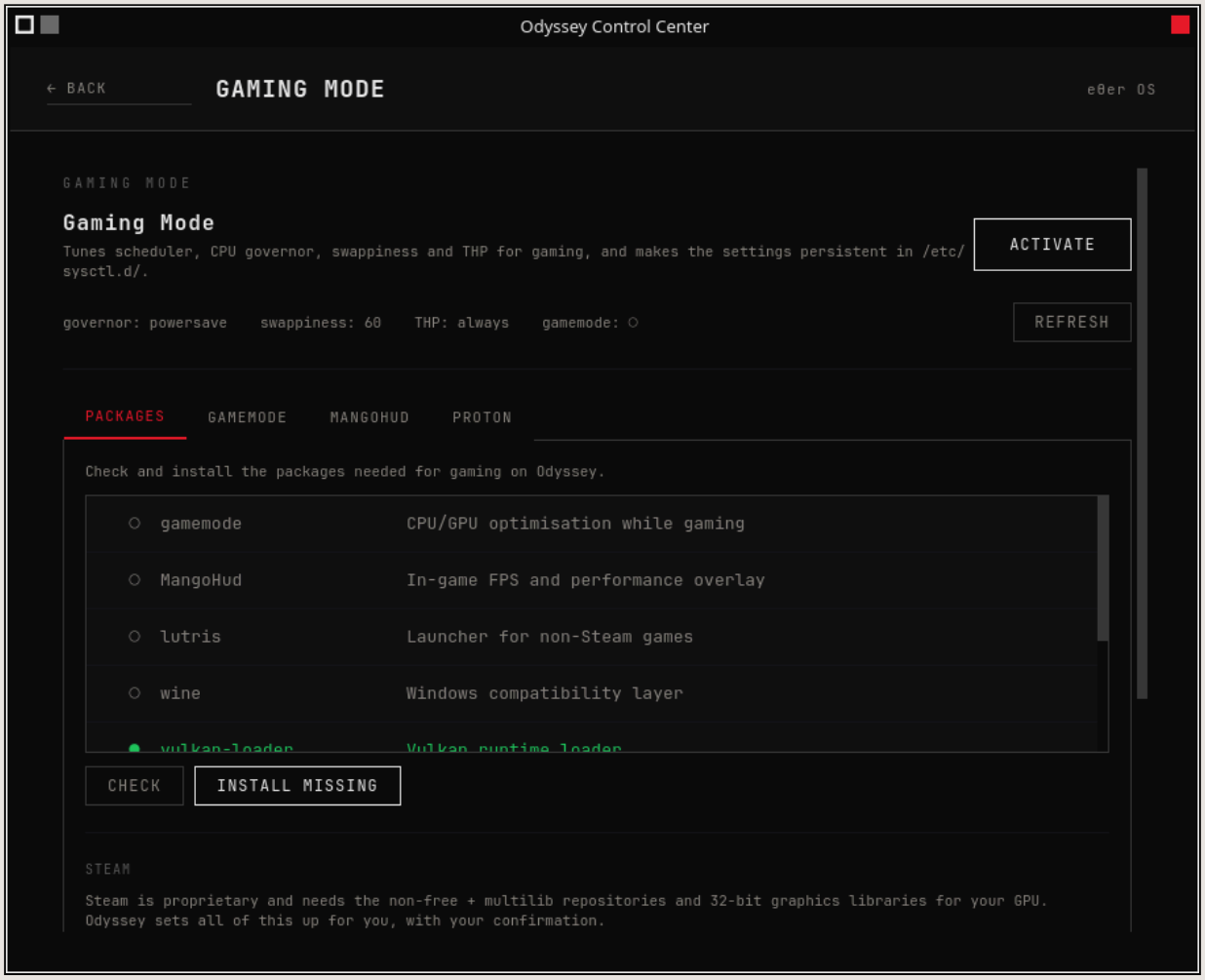

FIG · Верхний раздел панели: «Gaming Mode — Tunes scheduler, CPU governor, swappiness and THP for gaming, and makes the settings persistent in /etc/sysctl.d/», кнопка ACTIVATE, кнопка REFRESH, и строка статуса в реальном времени: governor powersave, swappiness 60, THP always, gamemode off.

Строка статуса показывает те же значения в реальном времени, что и панели Kernel и Memory — governor, swappiness, состояние THP, и работает ли сервис `gamemode` — так что вы можете подтвердить одним взглядом, действительно ли активация сработала, не посещая отдельно ни одну из двух панелей.

WHAT	Gaming Mode → ACTIVATE.
WHY	Вручную заходить в Kernel и Memory, чтобы вручную задать governor, swappiness и значение THP, подходящие для игр — это именно та многошаговая рутина, которую должен устранить выделенный режим.
RESULT	Scheduler, governor, swappiness и THP все переключаются на значения, подходящие для игр, применяются немедленно и становятся постоянными — строка статуса обновляется, отражая изменение.
RISK	Низкий, и обратимый — те же значения, которые вы бы иначе задали вручную в Kernel/Memory, просто объединённые в один клик.

9.2 Визуальное руководство — Packages и Steam

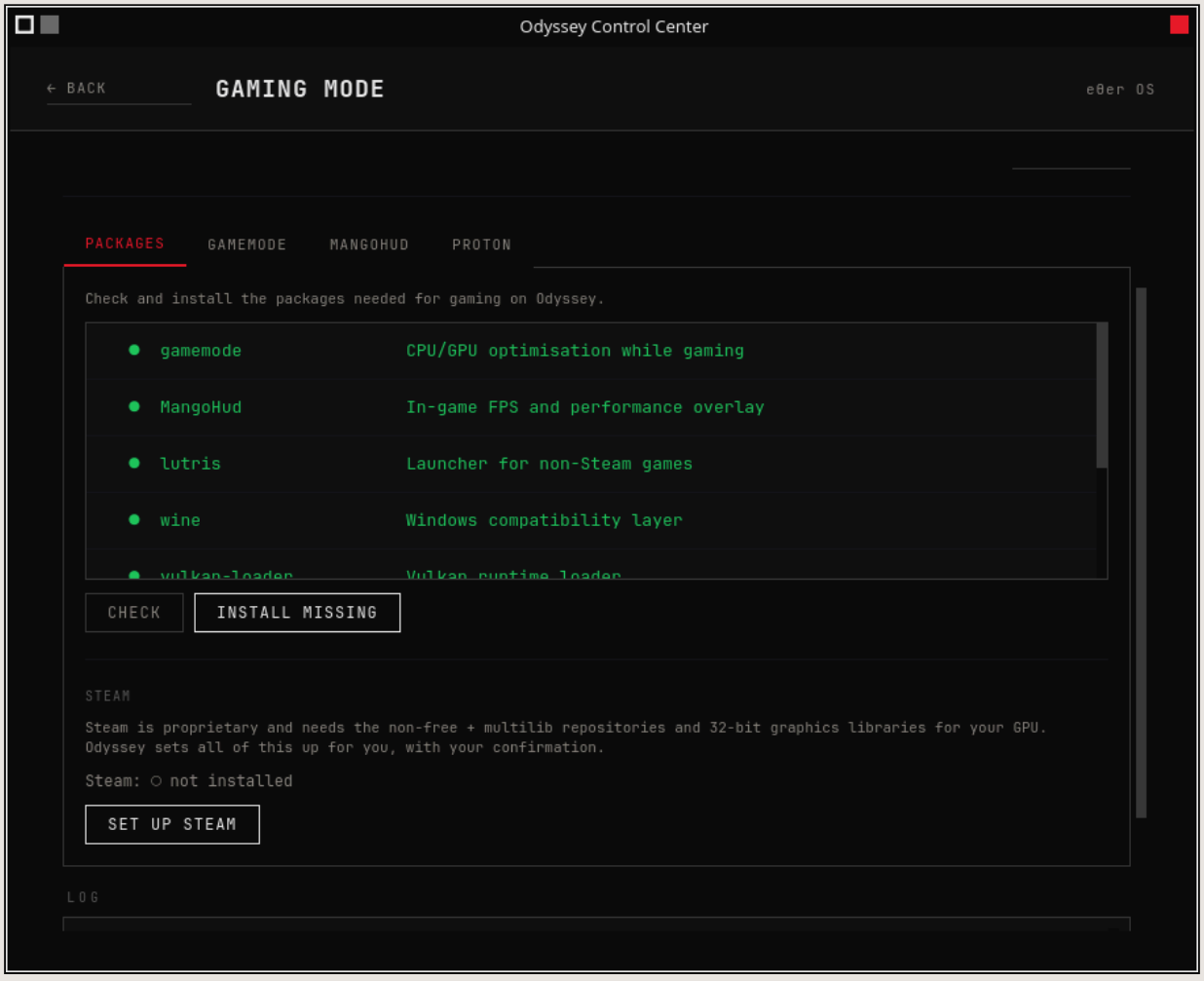

FIG · Вкладка PACKAGES: gamemode, MangoHud, lutris, wine, vulkan-loader — все с зелёными закрашенными точками (установлены); CHECK и INSTALL MISSING вверху; раздел Steam ниже с «Steam: ● installed» и кнопкой REINSTALL STEAM.

Список проверки пакетов, обычно необходимых для игр — CHECK показывает, чего не хватает, INSTALL MISSING восполняет пробелы за одно действие. Как только всё показывает закрашенную зелёную точку, как на скриншоте, вы полностью настроены.

У Steam есть собственный выделенный процесс под списком проверки, потому что ему нужно больше, чем простая установка пакета.

1 **SET UP STEAM**

Нажмите — это ещё ничего не устанавливает, сначала открывается информационное диалоговое окно подтверждения.

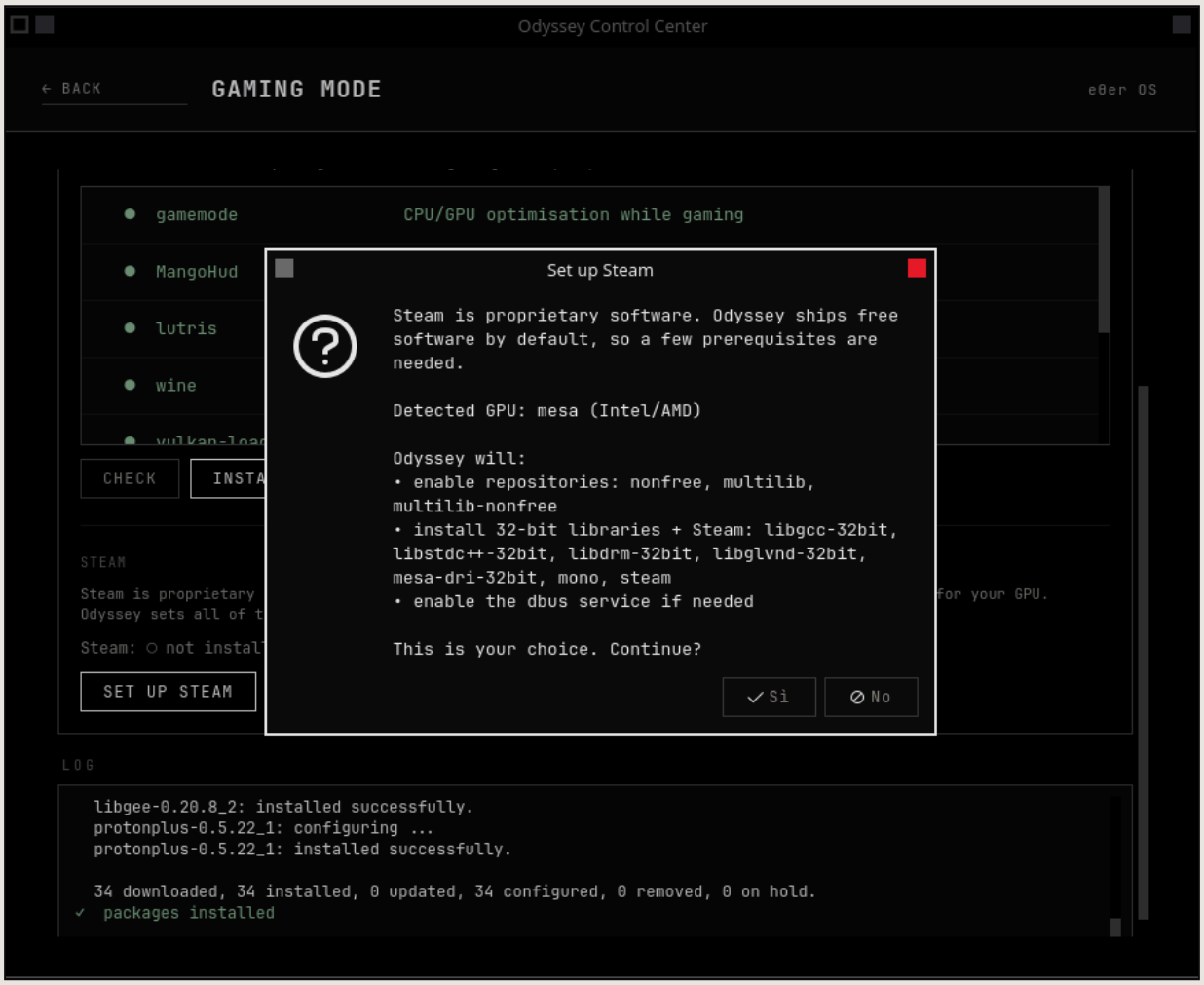

FIG · Диалоговое окно подтверждения: «Steam is proprietary software. Odyssey ships free software by default, so a few prerequisites are needed. Detected GPU: mesa (Intel/AMD).» Маркированный список того, что именно будет включено и установлено, заканчивающийся «This is your choice. Continue?» с кнопками Sí / No.

Диалоговое окно точно сообщает, что собирается произойти, прежде чем вы согласитесь: какие репозитории включаются (`nonfree`, `multilib`, `multilib-nonfree`), какие 32-битные библиотеки устанавливаются в соответствии с вашим **обнаруженным** GPU, и что сервис dbus будет включён, если ещё не включён. Ничего не скрыто за подтверждением.

2 **Sí**

Подтверждает. Журнал передаёт реальную установку — загрузки пакетов, настройку, завершение.

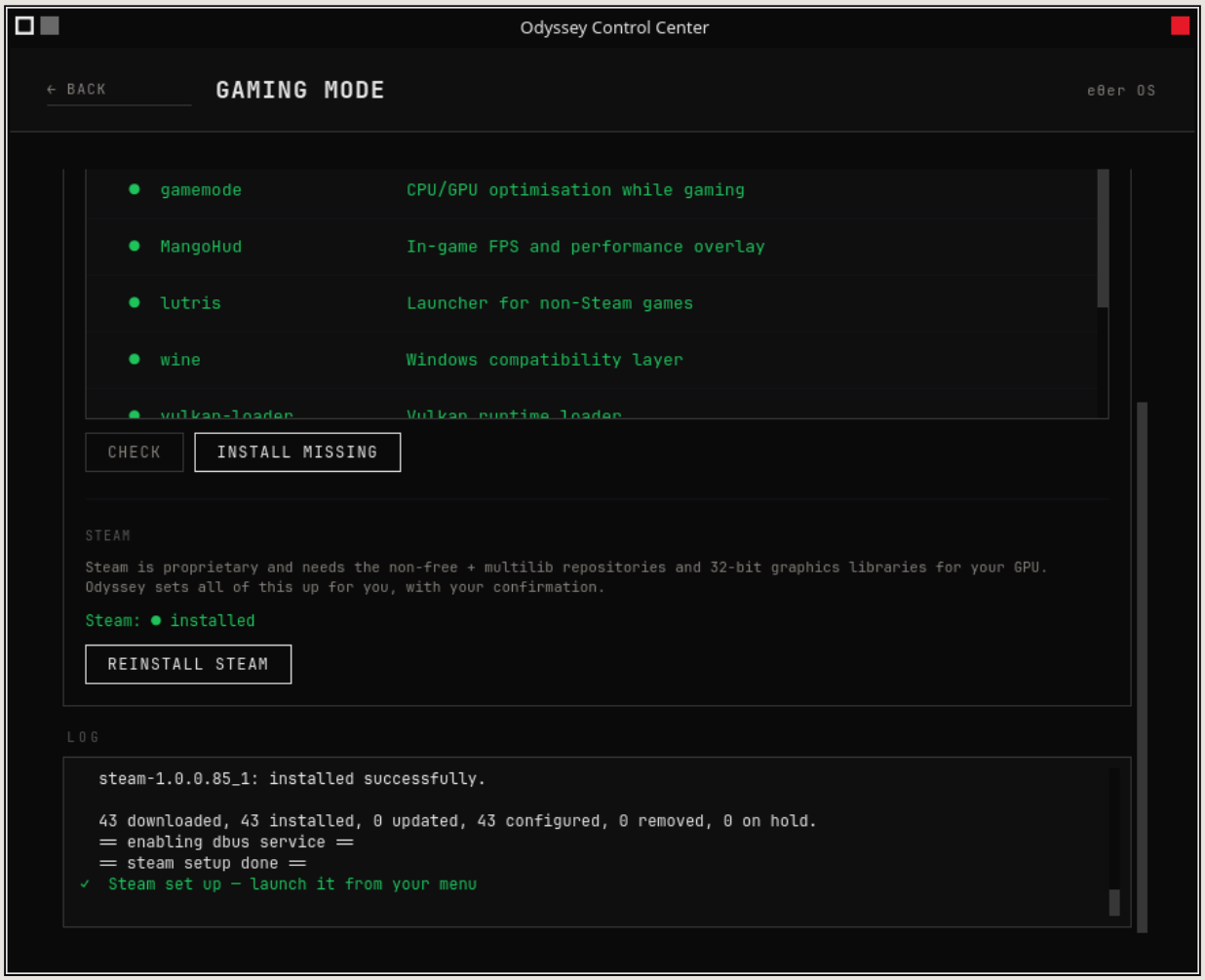

FIG · Вкладка PACKAGES после настройки: все пакеты зелёные, Steam показывает «● installed» с REINSTALL STEAM, а журнал заканчивается на «Steam set up — launch it from your menu.»

WHAT	Gaming Mode → вкладка PACKAGES → SET UP STEAM → проверьте диалоговое окно → Sí.
WHY	Steam проприетарен и нуждается в репозиториях non-free/multilib плюс 32-битных библиотеках, специфичных для GPU, которые Void не включает по умолчанию — правильно сделать это вручную — распространённое препятствие для новичков в системе на основе Void.
RESULT	Необходимые репозитории, сам Steam, и 32-битные графические библиотеки, соответствующие вашему конкретному GPU, все установлены — запустите Steam из меню приложений после этого.
RISK	Низкий — это только включает репозитории и добавляет пакеты; ничего существующего не удаляется и не переконфигурируется. Диалоговое окно показывает полный список того, что произойдёт, прежде чем вы подтвердите.

9.3 GameMode и MangoHud

GameMode (собственная вкладка) — это демон производительности для отдельных игр от Feral Interactive — игра, запущенная через него, запрашивает временное повышение производительности (governor CPU, приоритет I/O) только для своего собственного процесса,

автоматически возвращаемое, когда игра закрывается. **MangoHud** (собственная вкладка) — это накладной слой на экране, показывающий FPS, время кадра, нагрузку CPU/GPU и температуры во время игры. Обе вкладки предлагают элементы управления установкой/проверкой и, для MangoHud, быстрая настройка внешнего вида накладного слоя.

9.4 Proton

Управление слоем совместимости для запуска игр Windows через Steam — проверка того, что установлено и доступно, и установка дополнительных версий Proton, включая сборки сообщества вроде Proton-GE, когда конкретной игре нужна та, которую Valve не распространяет по умолчанию.

```
▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.
```

9.5 Advanced

ЧТО ЗАПИСЫВАЕТ ACTIVATE

Те же самые файлы, что и действия MAKE PERSISTENT панелей Kernel и Memory — Gaming Mode это пакет, а не отдельный механизм:

```
# значения sysctl, сделанные постоянными
$ cat /etc/sysctl.d/99-odyssey.conf
# CPU governor — это не sysctl, основан на sysfs, не делается постоянным
# этим механизмом; примените заново после загрузки, если вы на него полагаетесь
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

ЧТО НА САМОМ ДЕЛЕ ВЫПОЛНЯЕТ SET UP STEAM

Само диалоговое окно подтверждения — это почти полное техническое раскрытие, но конкретно, на системе Mesa (Intel/AMD):

```
$ sudo xbps-install -y void-repo-nonfree void-repo-multilib void-repo-multilib-nonfree
$ sudo xbps-install -Sy steam libgcc-32bit libstdc++-32bit libdrm-32bit libglvnd-32bit
mesa-dri-32bit mono
# системы NVIDIA дополнительно скачивают nvidia-libs-32bit вместо mesa-dri-32bit —
# панель определяет производителя вашего GPU и соответственно корректирует список пакетов
```

GAMEMODE И MANGOHUD ВРУЧНУЮ

```
# запустить любую игру с активным GameMode
$ gamemoderun ./game_binary
# или, для игры в Steam, параметры запуска:
$ gamemoderun %command%
# накладной слой MangoHud, параметры запуска:
$ mangohud %command%
# собственный конфигурационный файл MangoHud, для настройки того, что показывает накладной
слой
$ $EDITOR  /.config/MangoHud/MangoHud.conf
```

УСТАНОВКА КОНКРЕТНОЙ ВЕРСИИ PROTON-GE

Вкладка PROTON оборачивает `protonplus` или эквивалентный менеджер версий — ручной эквивалент — скачать релиз из `GloriousEggroll/proton-ge-custom` на GitHub и распаковать его в директорию инструментов совместимости Steam:

```
$ mkdir -p  /.steam/root/compatibilitytools.d
$ tar -xf GE-Proton-VERSION.tar.gz -C  /.steam/root/compatibilitytools.d
# затем выберите её для каждой игры в Steam: Properties → Compatibility
```

CHAPTER 10

Telemetry

Несмотря на название, эта панель — не настройка, которую вы конфигурируете, а монитор в реальном времени. Odyssey не собирает и не отправляет о вас ничего, поэтому здесь нечего отключать. Вместо этого данная панель показывает вам в реальном времени каждое исходящее соединение, которое устанавливает ваша **собственная** система — так что «с кем сейчас разговаривает моя машина» — это то, на что можно ответить, посмотрев, а не поверив утверждению. Она называет себя **Reverse Telemetry** именно поэтому: она переворачивает наблюдение, направляя его на саму систему, от вашего имени.

10.1 Визуальное руководство

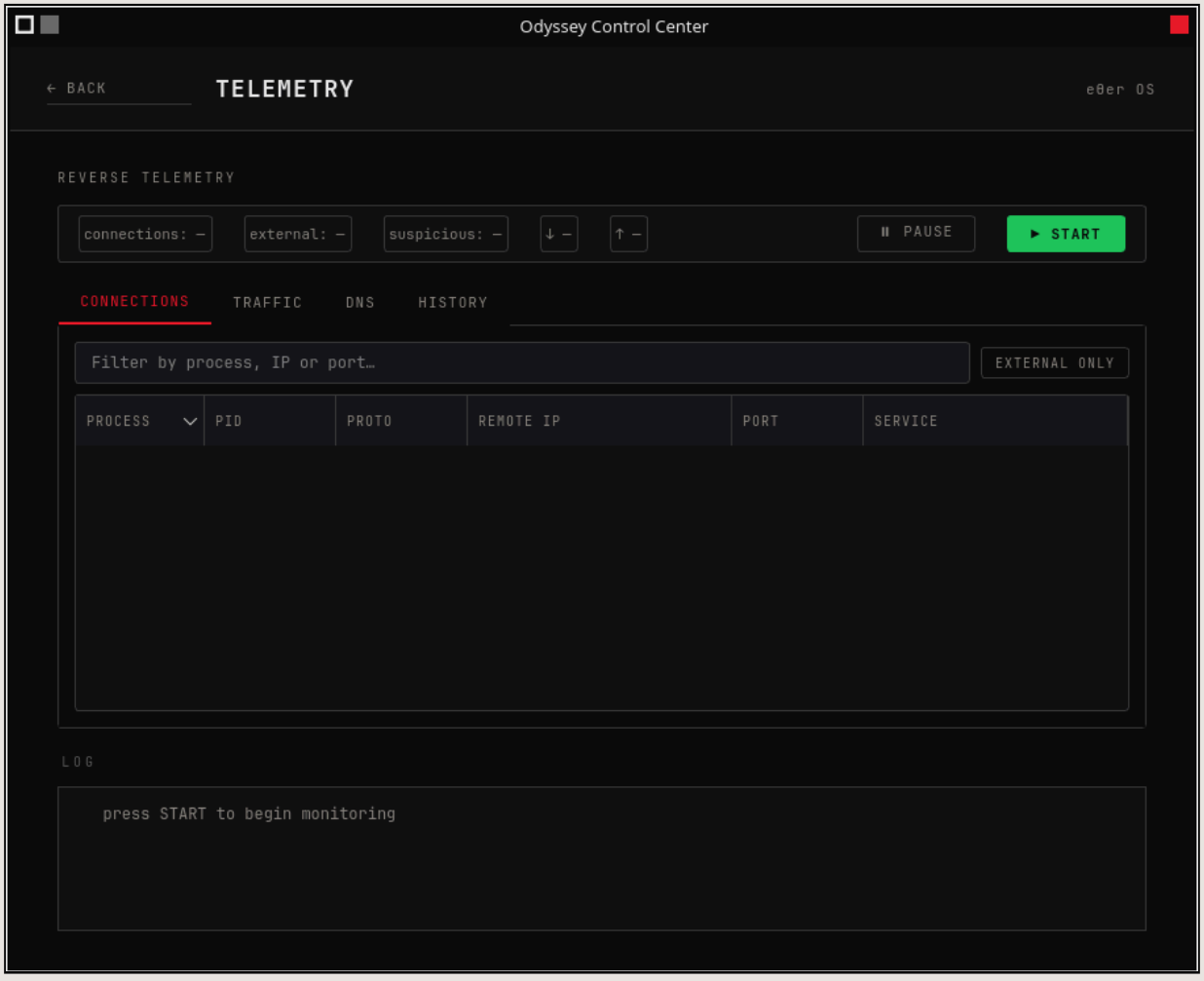

FIG · До запуска: connections/external/suspicious показывают всё «—», зелёная кнопка START, и журнал говорит «press START to begin monitoring». Четыре вкладки — CONNECTIONS, TRAFFIC, DNS, HISTORY — остаются пустыми над незаполненной таблицей.

1 START

Запускает мониторинг в реальном времени — строка статуса и активная вкладка заполняются немедленно.

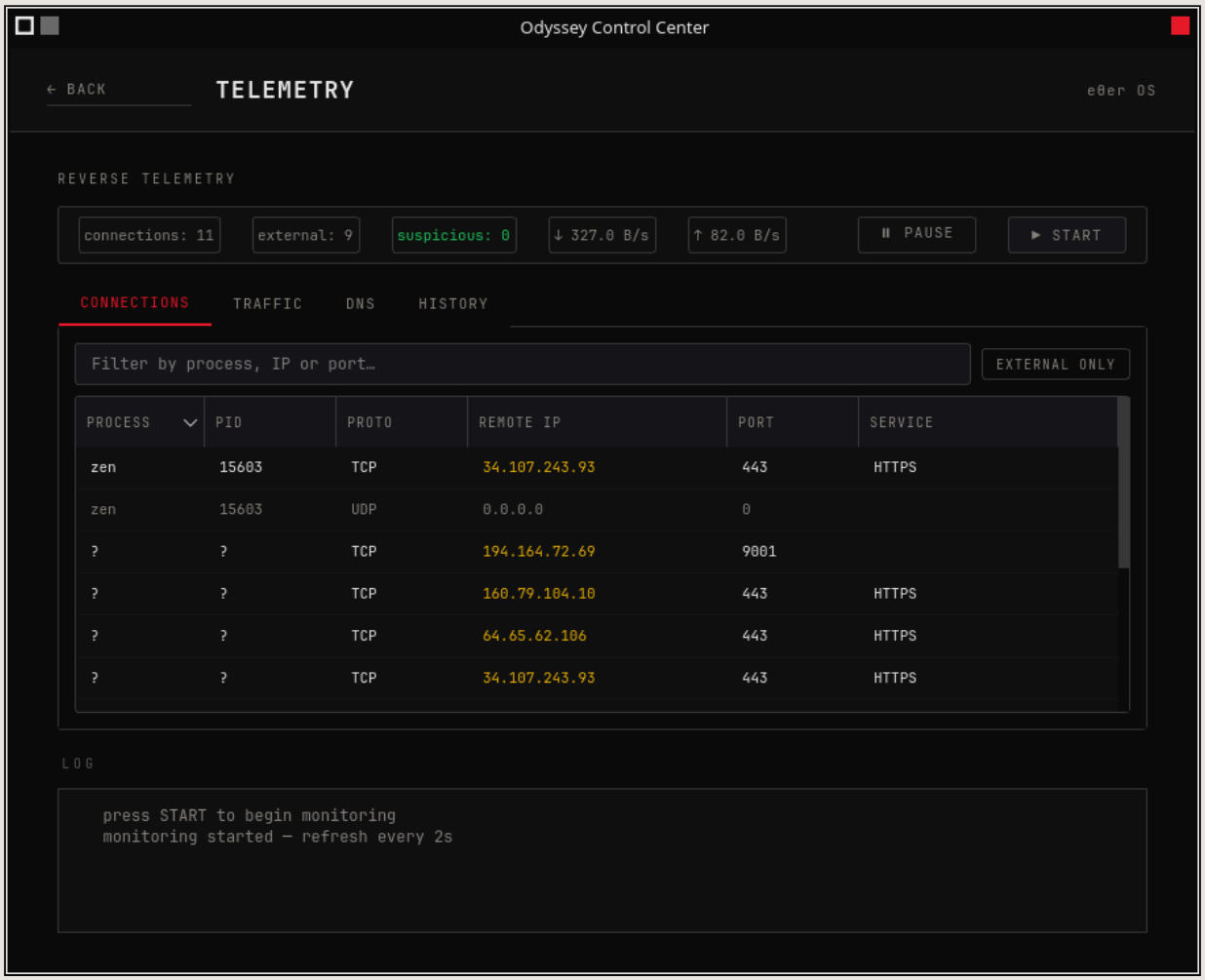

FIG · После запуска: «connections: 11 · external: 9 · suspicious: 0 · ↓327.0 B/s · ↑82.0 B/s», PAUSE теперь доступна, и таблица CONNECTIONS заполнена реальными строками: процесс «zen» по TCP к 34.107.243.93:443 (HTTPS), несколько строк отмечены «?» для процесса/PID с соединениями к другим адресам.

Строка статуса вверху предоставляет суммарные показатели в реальном времени — соединения, внешние (против локальных), подозрительные, текущие скорости загрузки/отправки — и **PAUSE** замораживает представление, не теряя захваченное, **START** возобновляет его.

ЧЕТЫРЕ ВКЛАДКИ

ВКЛАДКА	ЧТО ПОКАЗЫВАЕТ
CONNECTIONS	Одна строка на активное соединение: процесс, PID, протокол, удалённый адрес, порт, и служба, известная для этого порта (443 → HTTPS, 22 → SSH). Сортируется по любому столбцу.
TRAFFIC	Пропускная способность по интерфейсу — текущая скорость загрузки/отправки, плюс накопленные итоги с начала мониторинга.
DNS	Поток разрешений доменных имён в реальном времени по мере их появления — часто понятнее голого IP, поскольку имя хоста говорит вам **что**, а не только **куда**.

ВКЛАДКА	ЧТО ПОКАЗЫВАЕТ
HISTORY	Всё, что захватила сессия, в виде прокручиваемого журнала — полезно для того, чтобы поймать что-то краткое, что к моменту вашего взгляда уже не является активным соединением.

Строки, где процесс показывает `?` (видны на скриншоте), означают, что соединение было захвачено, но исходный процесс не удалось разрешить по PID в тот момент — часто встречается у очень кратковременных соединений. Поле **фильтра** над таблицей сужает по процессу, IP или порту, а **EXTERNAL ONLY** скрывает чисто локальный трафик.

WHAT	Telemetry → START → наблюдайте за CONNECTIONS, или сортируйте по PROCESS или REMOTE IP, чтобы расследовать что-то конкретное.
WHY	Это прямой и конкретный ответ на вопрос «есть ли на этой машине что-то, достигающее сети, чего не должно быть» — читается напрямую из собственных таблиц соединений ядра, а не из утверждения поставщика о собственном программном обеспечении.
RESULT	Живая и точная картина каждого исходящего соединения, непрерывно обновляемая, пока работает мониторинг и не находится на паузе.
RISK	Никакого — полностью только для чтения.

Соединения от процессов без обычной причины самостоятельно обращаться к сети — голый shell, независимо запущенный интерпретатор скриптов — автоматически отмечаются как «suspicious» в строке статуса, так что аномалия выделяется, а не теряется в обычном трафике вашего браузера и системных служб.

```
▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.
```

10.2 Advanced

ОТКУДА БЕРУТСЯ ДАННЫЕ

Панель читает те же таблицы соединений на уровне ядра, которые читал бы любой стандартный инструмент, обновляемые с коротким интервалом, пока мониторинг активен — без какого-либо собственного специального драйвера или модуля ядра. Эквиваленты командной строки:

```
# каждый открытый сокет, с процессом-владельцем — те же поля, что и в CONNECTIONS
$ sudo ss -tupn
# пропускная способность по интерфейсу — те же числа, что и в TRAFFIC
$ ip -s link
# разовое представление в реальном времени, если хотите увидеть его вне панели
$ sudo nethogs
```

РАЗРЕШЕНИЯ DNS, ВРУЧНУЮ

```
# захват исходящих DNS-запросов в реальном времени (нужен инструмент захвата пакетов)
$ sudo tcpdump -i any -n port 53
```

ПОЧЕМУ НЕКОТОРЫЕ СТРОКИ ПОКАЗЫВАЮТ PROCESS КАК «?»

Сопоставление живого соединения с владеющим им процессом требует чтения `/proc/PID/fd` для каждого кандидата в процессы в момент захвата — если соединение открывается и закрывается быстрее, чем интервал обновления панели, или владеющий процесс уже завершился, у панели нет окна для этого сопоставления, и она показывает `?` вместо того, чтобы гадать. Это ограничение по времени, а не признак того, что что-то скрывается.

CHAPTER 11

About

Единый исчерпывающий диагностический лист машины, обновляемый каждый раз при открытии вкладки, чтобы всегда отражать то, что изменилось в других панелях, без необходимости заново открывать Control Center.

11.1 Визуальное руководство

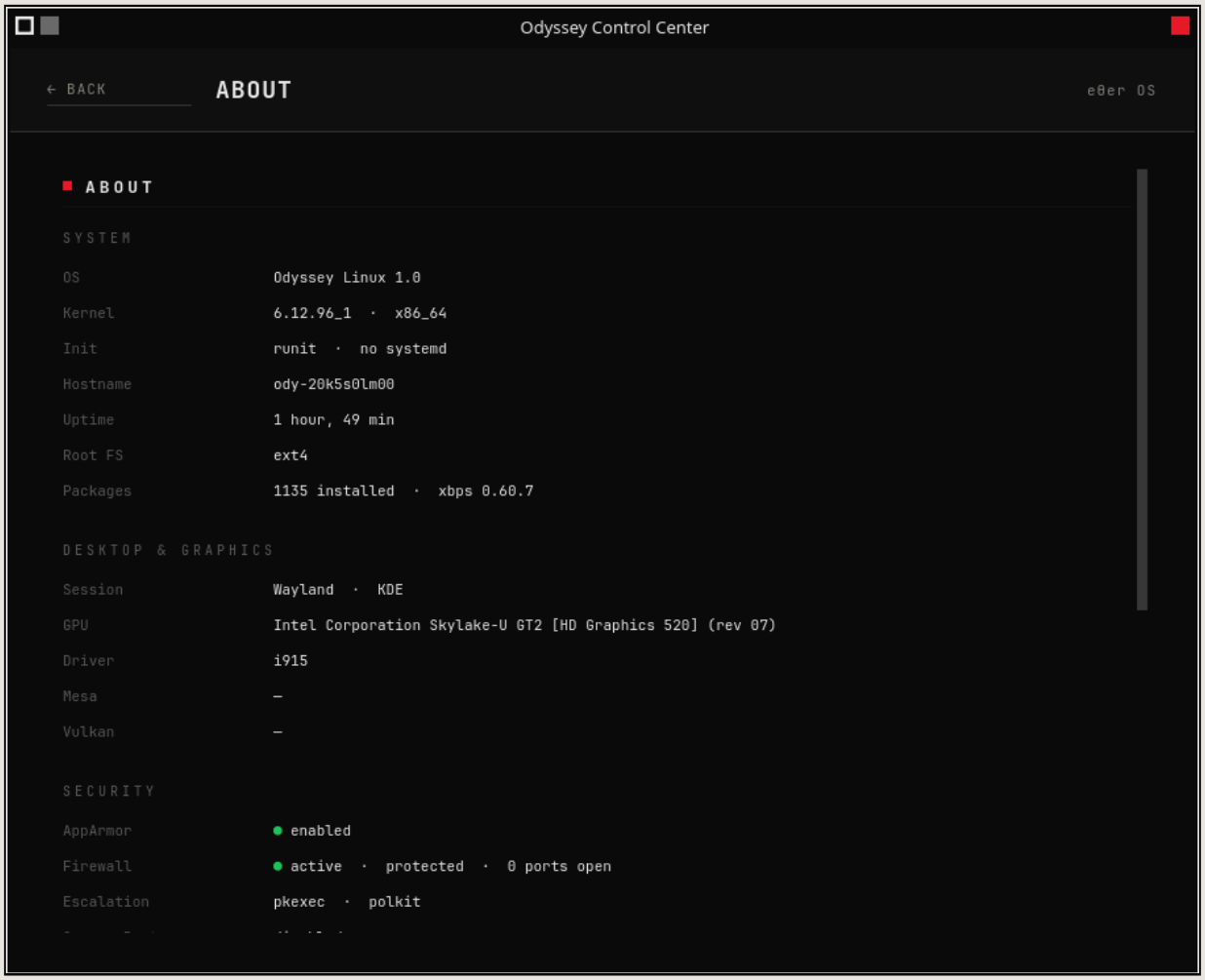

FIG · Панель прокручена до самого верха: раздел SYSTEM показывает OS «Odyssey Linux 1.0», Kernel 6.12.96_1 · x86_64, Init runit · no systemd, Hostname, Uptime, Root FS ext4, Packages «1135 installed · xbps 0.60.7»; DESKTOP & GRAPHICS ниже с Session, GPU, Driver, Mesa, Vulkan; SECURITY начинает показывать AppArmor «● enabled» и Firewall «● active · protected · 0 ports open».

Три раздела, по порядку: **SYSTEM** (ядро, init, имя хоста, время работы, файловая система, количество пакетов), **DESKTOP & GRAPHICS** (тип сессии, GPU, драйвер, поддержка Mesa/Vulkan),

и ниже **SECURITY** (статус AppArmor и Firewall с цветными индикаторами, а не только текстом — зелёный для включено/активно) и **HARDWARE** (материнская плата, аудио).

WHAT	About → прокрутите три раздела, чтобы проверить любое значение — версию ядра, используемый драйвер, действительно ли активны AppArmor и файрвол.
WHY	Это самый быстрый способ подтвердить состояние всей системы в одном месте, не открывая пять разных панелей.
RESULT	Полный, актуальный снимок — обновляемый при каждом посещении вкладки, поэтому он никогда не показывает устаревшую информацию из предыдущей сессии.
RISK	Никакого — полностью только для чтения.

11.2 Copy report

Одна кнопка вверху панели, **COPY REPORT**, помещает весь лист в буфер обмена как обычный текст — каждый раздел, каждое значение — готовый для вставки прямо в сообщение на форуме или в отчёт в баг-трекере.

WHAT	About → COPY REPORT → вставьте в отчёт об ошибке или сообщение на форуме.
WHY	Отчёты об ошибках полезны настолько, насколько полезны приложенные к ним сведения о системе, а вручную набирать версию ядра, драйвер, статус AppArmor и десяток других полей — это именно то трение, из-за которого многие пропускают этот шаг.
RESULT	Полный, правильно отформатированный отчёт о системе в буфере обмена, готовый к вставке — та же информация, которую иначе сопровождающему пришлось бы запрашивать у вас по полю.
RISK	Никакого для вашей системы. Отчёт — это диагностическая информация (версии, состояния, модель оборудования) — он не включает личные файлы или учётные данные — но, как и с любым диагностическим дампом, перечитайте его перед публикацией где-то публично, если не уверены, что он содержит.

▸ ADVANCED

Past this point: config file paths, raw commands, and what the panel is doing underneath.

11.3 Advanced

ОТКУДА БЕРЁТСЯ КАЖДОЕ ПОЛЕ

Ничто в этой панели не выдумано и не кэшировано с момента установки — каждое значение запрашивается вживую, каждый раз при открытии вкладки:

```
$ uname -r # Kernel
$ cat /proc/sys/kernel/osrelease
$ hostnamectl 2>/dev/null || hostname
$ uptime -p
$ xbps-query -l | wc -l # Packages installed
$ xbps-query -v # xbps version
$ lspci -k | grep -A3 VGA # GPU + Driver
$ glxinfo | grep “OpenGL version” # Mesa
$ vulkaninfo –summary # Vulkan
```

Раздел Security в частности:

```
$ sudo aa-status –enabled && echo enabled
$ sudo nft list ruleset | head -1 # firewall active/mode
```

ПОЧЕМУ СТОИТ ПРОВЕРЯТЬ ABOUT В ПЕРВУЮ ОЧЕРЕДЬ

Поскольку панель заново опрашивает систему при каждом посещении, About — самый быстрый способ подтвердить, что изменение, сделанное в другой панели, действительно возымело эффект — примените режим файрвола в Firewall, затем проверьте раздел Security в About, вместо того чтобы снова открывать сам Firewall. Она построена как перекрёстная проверка «действительно ли это сработало?» для остальной части Control Center.

CHAPTER 12

NVIDIA driver

ОЖИДАЕТ ИНТЕГРАЦИИ

Эта глава — заполнитель. Модуль NVIDIA появляется в Control Center только на машинах с обнаруженным оборудованием NVIDIA, и скриншоты для него можно сделать только на такой машине — они будут предоставлены отдельно, и эта глава будет завершена в будущей редакции этого руководства. Она намеренно оставлена последней, после About, именно для того, чтобы добавление её позже не сдвигало нумерацию страниц или ссылки на разделы ни в одной другой главе.

Что уже известно и что закрепит завершённую главу: панель несёт видимый значок **BETA** и предваряющее примечание о том, что сам драйвер проприетарен — упакован Void из официального установщика NVIDIA с зафиксированной контрольной суммой, но не подлежит независимой проверке — та же честность, которую остальной Odyssey применяет к себе самому. Она определяет вашу конкретную GPU, сверяясь со встроенной базой данных, и автоматически выбирает правильную ветку драйвера (current/Turing-и-новее, или одну из трёх устаревших ветвей для более старых карт, переходя на драйвер с открытым исходным кодом `nouveau`, когда ничего проприетарного не подходит). Установка идёт как отслеживаемый конвейер из семи шагов с видимым индикатором прогресса, и панель заранее выводит собственную команду восстановления — `odyssey-nvidia --revert` из текстовой консоли — ещё до того, как вы нажмёте установить. На гибридных ноутбуках автоматически появляется раздел PRIME render-offload, позволяющий конкретной игре или программе работать на GPU NVIDIA, пока всё остальное остаётся на встроенной GPU ради автономности батареи.