ODYSSEY LINUX · REFERENZDOKUMENTATION

Odyssey Control Center — Praxis-handbuch

Jedes Panel, Abschnitt für Abschnitt. Was es tut, was es ändert,
und worauf zu klicken ist — geschrieben für alle, die es noch nie benutzt haben.

Launch Edition

Don’t trust me. Verify me.

— nobody

Odyssey Linux — Odyssey Control Center: Praxishandbuch
Launch Edition. Deckt das Control Center so ab, wie es auf der ISO der Launch Edition ausgeliefert wird — zehn vollständige Panels, von Boot bis About, plus ein Platzhalterkapitel für das NVIDIA-Treibermodul (nur auf NVIDIA-Hardware sichtbar, wartet auf Screenshots von einer Maschine, die eine besitzt).

Copyright © 2026 the Odyssey Linux project.
Veröffentlicht unter der Lizenz Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).

Dieses Handbuch ist die praktische Ergänzung zum Security and Hardening Guide, der die Panels Firewall und AppArmor in voller technischer Tiefe behandelt — hier findest du die visuelle Schritt-für-Schritt-Anleitung und einen Verweis für den Rest.

Jeder Bildschirm, jede Schaltfläche und jedes Verhalten, das hier beschrieben wird, wurde aus dem tatsächlichen Quellcode des Control Center und aus echten Screenshots gelesen, nicht aus dem Gedächtnis.

Schaltflächenbeschriftungen, Panel-/Tab-Namen, Dateipfade und Befehle bleiben auf Englisch, weil sie die tatsächliche Benutzeroberfläche des Programms sind: Sie übersetzt im Control Center zu suchen, würde sie nicht finden.

INHALT

CHAPTER 1

Was das Control Center ist

Das Odyssey Control Center sind zehn kleine, unabhängige Programme hinter einem einzigen Launcher — eines pro Systembereich: Boot, Kernel, Memory & Swap, Firewall, AppArmor, Privacy, Containers, Gaming Mode, Telemetry, About, und (auf NVIDIA-Hardware) das NVIDIA-Treibermodul.

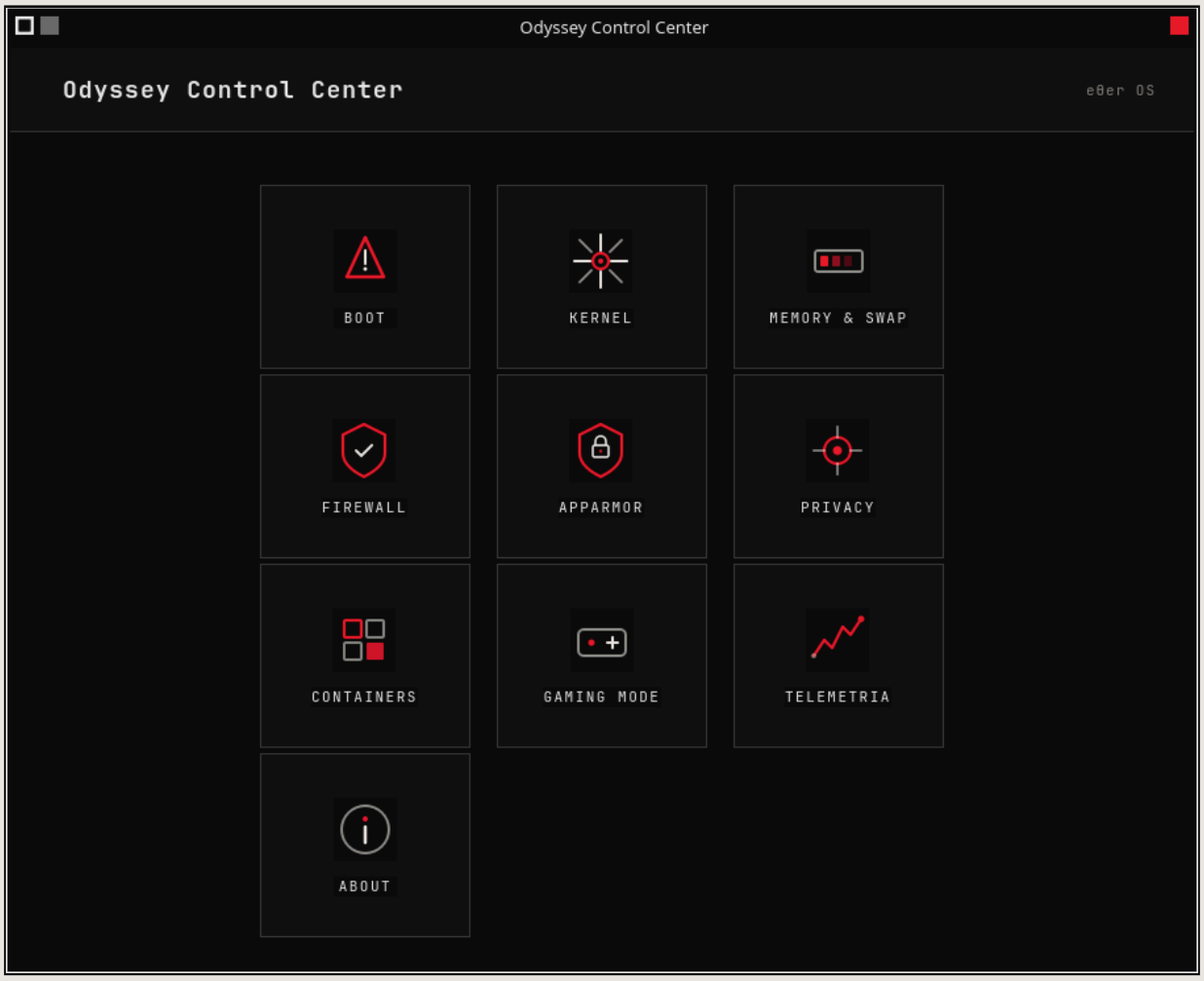

FIG · Der Startbildschirm des Control Center — eine Kachel pro Panel.

Jedes Panel ist eine dünne grafische Schicht über dem eigentlichen Linux-Mechanismus darunter: der GRUB-Konfiguration, der sysctl- Schnittstelle des Kernels selbst, `nftables`, `podman`. Ein Panel erfindet nie einen eigenen privaten Zustand — es liest die tatsächlichen Dateien und Befehle, zeigt sie dir, und schreibt genau das, worum du gebeten hast. Nichts, was es tut, ist verborgen, und alles, was es tut, lässt sich auch von Hand erledigen, im Terminal, mit den genauen Befehlen, die dieses Handbuch neben jeder Schaltfläche zeigt.

Dies ist ein Praxishandbuch, keine Referenz für jede Option — es beantwortet „Was klicke ich, und was passiert dann?“ Jedes Kapitel hat denselben zweiteiligen Aufbau: zuerst eine **visuelle Anleitung**, Screenshot für Screenshot, Schaltfläche für Schaltfläche, geschrieben für alle, die das Control Center noch nie geöffnet haben — dann einen **ADVANCED**-Abschnitt, gekennzeichnet durch ein schwarzes

Banner, der ins Detail geht: exakte Dateipfade, die tatsächlichen Befehle, die jede Schaltfläche im Hintergrund ausführt, und was im Terminal zu tun ist, wenn du das Panel ganz überspringen willst. Lies nur die visuelle Anleitung, oder springe direkt zu Advanced, wenn du Linux bereits kennst — beide sind für sich genommen vollständig.

Für die beiden Panels mit echten Sicherheitsauswirkungen — Firewall und AppArmor — ist die vertiefte Referenz der **Security and Hardening Guide**; hier findest du die visuelle Anleitung, das Wesentliche, und einen Verweis.

1.1 Muster, die dir in jedem Panel begegnen

Ein paar Konventionen wiederholen sich in allen zehn Panels — sie einmal zu lernen gilt für alle.

DIE PASSWORTABFRAGE, NUR EINMAL PRO PANEL

Die meisten Panels müssen Dateien lesen oder schreiben, die root gehören — die GRUB-Konfiguration, sysctl, das Firewall-Regelwerk. Beim ersten Mal, wenn ein Panel das braucht, erhältst du einen einzigen Authentifizierungsdialog. Panels sind so gebaut, dass sie ihre Lesevorgänge bündeln: Boot zu öffnen liest zum Beispiel die GRUB-Konfiguration **und** den Zustand der EFI-Firmware in einer einzigen Anfrage, sodass du dein Passwort einmal eingibst, nicht zweimal. Ein Panel in derselben Sitzung erneut zu öffnen, fragt nicht noch einmal danach; nachfolgende Aktionen (eine Änderung anwenden, neu laden) zeigen ihren eigenen Fortschritt im Protokoll des Panels selbst.

DIE SCHRITT-ANZEIGE

Panels, die dich durch die Installation von etwas führen, bevor es nutzbar wird — os-prober in Boot, SearXNG und Tor in Privacy, die Container-Engine in Containers — zeigen oben eine kleine nummerierte Anzeige: eine Reihe von Schritten, der aktuelle hervorgehoben, und eine einzige Aktionsschaltfläche, die immer **den nächsten Schritt** ausführt, welcher es auch sei. Du musst nie raten, was zuerst zu klicken ist.

DAS PROTOKOLL AM UNTEREN RAND

Jedes Panel endet in einem kleinen scrollbaren Protokoll. Das ist keine Dekoration — es ist der exakte Befehl, der ausgeführt wurde, seine Ausgabe, und ein farbiges Ergebnis (grün für Erfolg, rot für Fehlschlag). Wenn ein Panel etwas tut, das du nicht verstehst, steht im Protokoll die eigentliche Antwort.

TRY VS. MAKE PERSISTENT

Ein wiederkehrendes Schaltflächenpaar, vor allem in Kernel und Memory: **TRY NOW** (oder **APPLY NOW**) ändert das laufende System sofort und setzt beim nächsten Neustart zurück, wenn du nichts weiter tust. **MAKE PERSISTENT** schreibt denselben Wert in eine Konfigurationsdatei unter `/etc/`, damit er auch Neustarts übersteht. Diese Trennung gibt es absichtlich — einen Wert auszuprobieren kostet nichts, und nichts ist dauerhaft, bis du es zweimal gesagt hast.

CHAPTER 2

Boot

Das Panel Boot hat zwei Tabs auf jeder Maschine, die im UEFI-Modus startet: **GRUB** (das eigentliche Boot-Menü) und **EFI** (die Firmware-Ebene darunter). Auf älteren, reinen BIOS-Maschinen gibt es nur GRUB, weil die EFI-Schicht darunter schlicht nicht existiert.

> ZWEI EBENEN, NICHT EINE
>
> **GRUB** ist das Boot-Menü, das du auf dem Bildschirm siehst — es entscheidet, welches Betriebssystem startet, sobald GRUB selbst bereits läuft. Die **EFI/UEFI-Firmware** liegt eine Ebene unter GRUB — sie entscheidet, an welchen Bootloader die Maschine überhaupt zuerst die Kontrolle übergibt, noch bevor GRUB geladen wurde. Die meisten Leute berühren den EFI-Tab nie; er existiert für die Fälle, die GRUB nicht erreichen kann, wie eine zweite Windows-Installation, die ihren eigenen Booteintrag unabhängig verwaltet.

2.1 Visuelle Anleitung — ein weiteres Betriebssystem finden und den Standard festlegen

Der GRUB-Tab öffnet sich im Abschnitt **Other OS detection**. Odyssey sucht standardmäßig nicht nach anderen Systemen — die meisten Installationen haben nur Odyssey auf der Festplatte, und ein Scan kostet bei jeder Neuerstellung des Boot-Menüs Zeit, ohne Nutzen. Ihn zu aktivieren ist eine geführte Sequenz aus drei Schritten, dargestellt als kleine nummerierte Anzeige.

1. **Installed**

   Das Scan-Werkzeug os-prober ist noch nicht auf dem System. Eine einzige Schaltfläche: **INSTALL OS-PROBER**.

2. **Enable scan**

   Das Werkzeug ist installiert, aber GRUB ist noch so eingestellt, es zu ignorieren — Odysseys Standard. Eine einzige Schaltfläche: **ENABLE SCAN**.

3. **Scan**

   GRUB wird es nun verwenden. Eine einzige Schaltfläche, die du auch später wieder benutzt, wann immer du ein weiteres System hinzufügst oder entfernst: **SCAN & REGENERATE**.

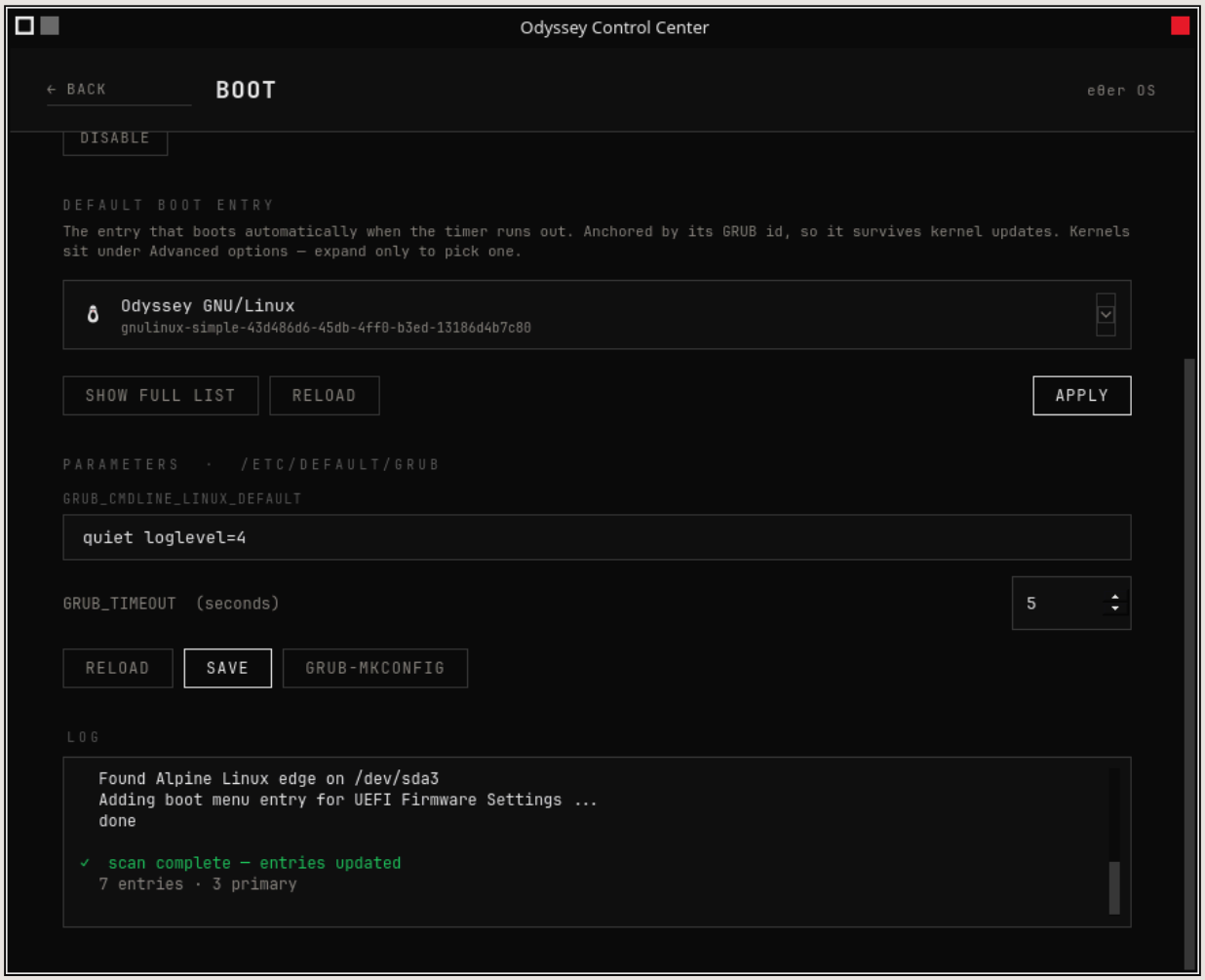

FIG · Alle drei Schritte grün — os-prober installiert, Scan aktiviert, bereit. Die einzelne Aktionsschaltfläche entspricht immer dem aktuellen Schritt.

Klicke durch die Schritte (jeder verlangt die Authentifizierung nur erneut, wenn nötig) und führe den Scan aus. Das Panel zeigt dir genau, was es gefunden hat, bevor du den Abschnitt verlässt:

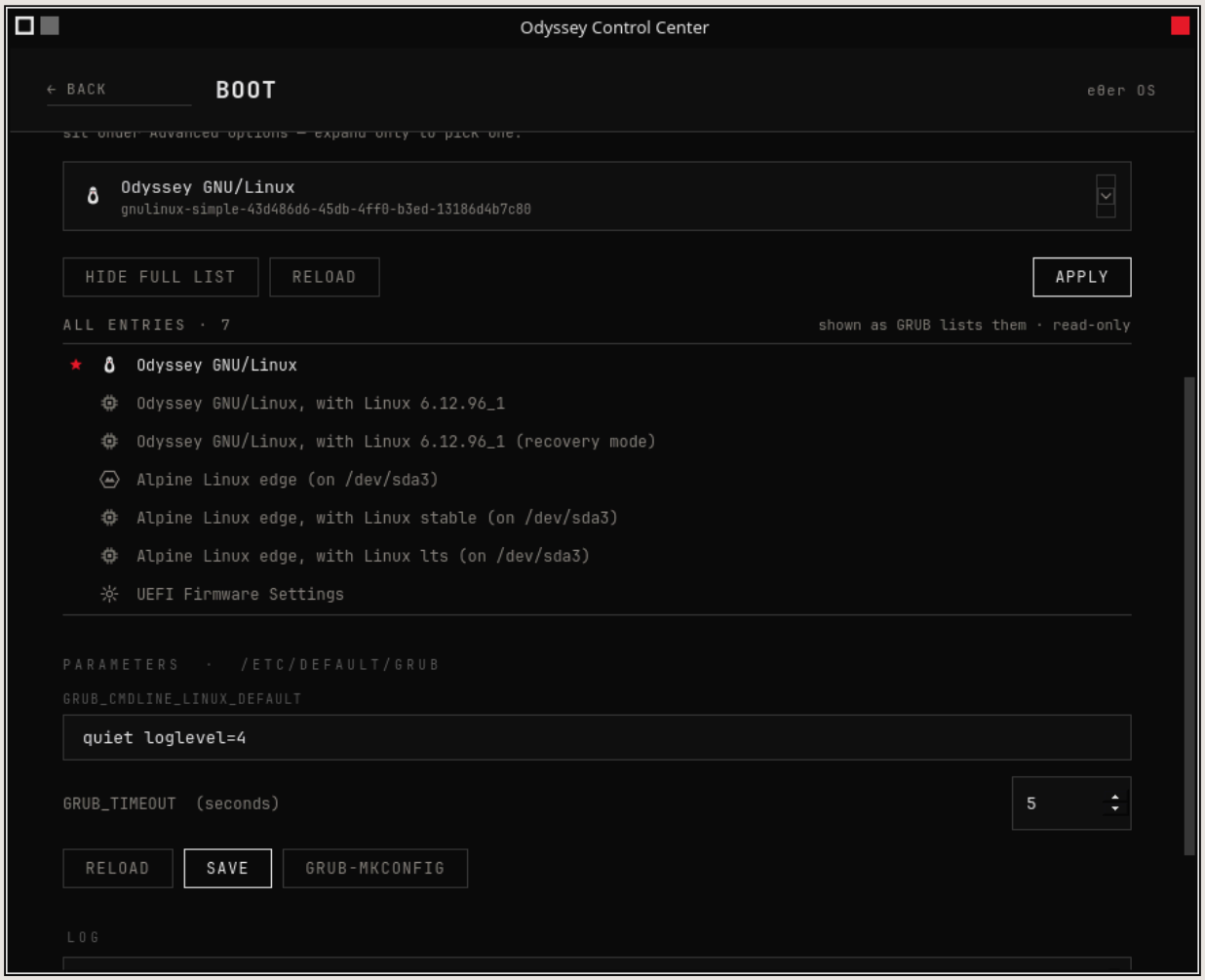

FIG · Scan abgeschlossen: Alpine Linux gefunden auf /dev/sda3, zusammen mit Odysseys eigenen Einträgen und dem Eintrag UEFI Firmware Settings — 7 Einträge, 3 primäre.

Das Protokoll bestätigt es in einfachen Worten — **„scan complete — entries updated · 7 entries · 3 primary.“** Alles, was os-prober jetzt findet, erscheint automatisch im Abschnitt darunter, bereit, gewählt zu werden.

2.2 Visuelle Anleitung — auswählen, was standardmäßig startet

Scrolle zu **Default boot entry**. Sie zeigt den aktuellen Standard als eine Zeile; ein Klick darauf öffnet ein Dropdown mit allem, was du stattdessen wählen kannst.

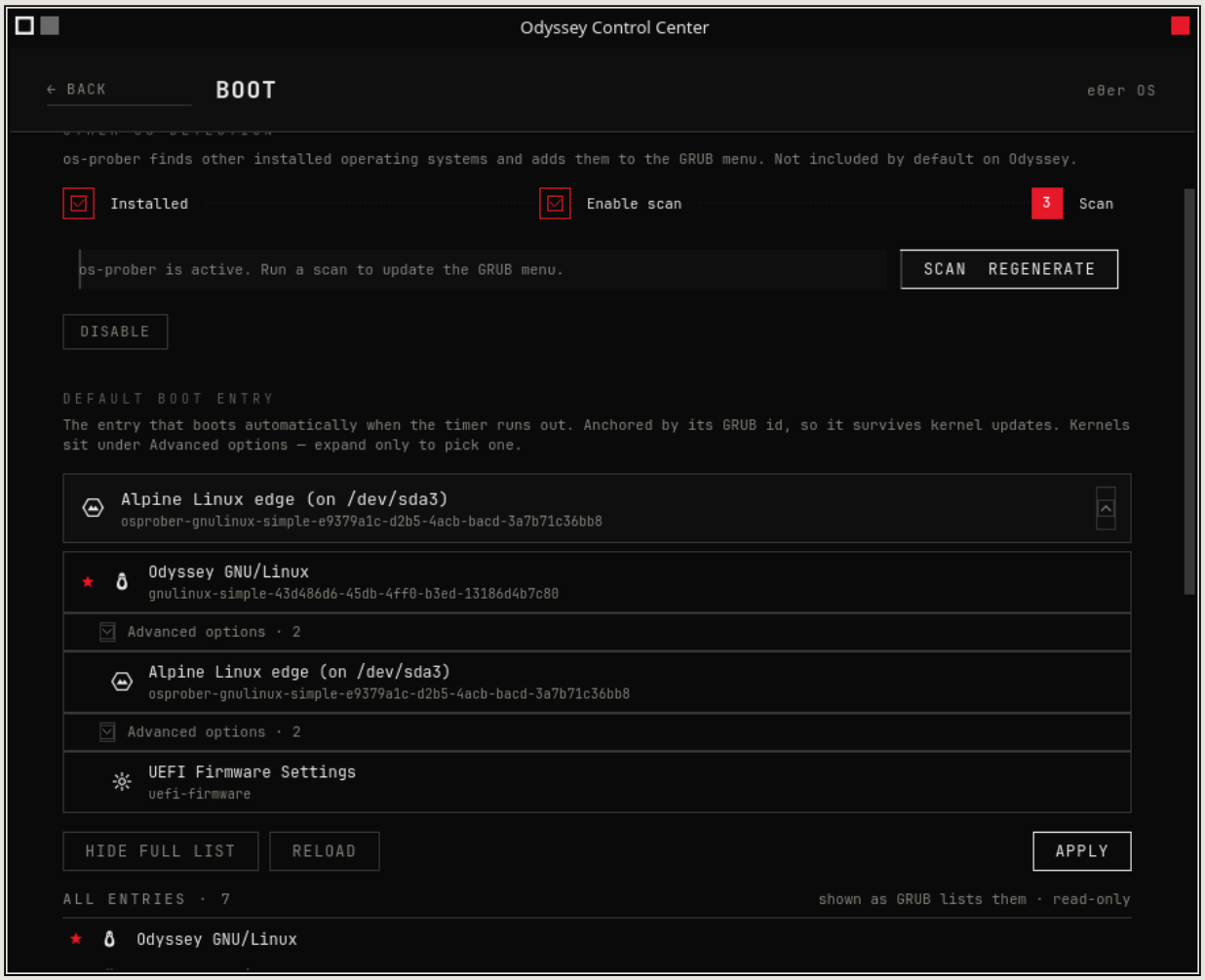

FIG · Das geöffnete Dropdown, mit Alpine Linux als ausstehender Auswahl markiert — noch nicht angewendet.

Einen Eintrag aus der Liste zu wählen **stellt die Wahl zurück** — sie hat noch keine Wirkung. Man sieht, dass sie nur zurückgestellt ist, weil das Feld weiterhin die neue Auswahl zeigt, aber nichts wurde geschrieben. Die Schaltfläche, die sie bestätigt, ist **APPLY**, rechts:

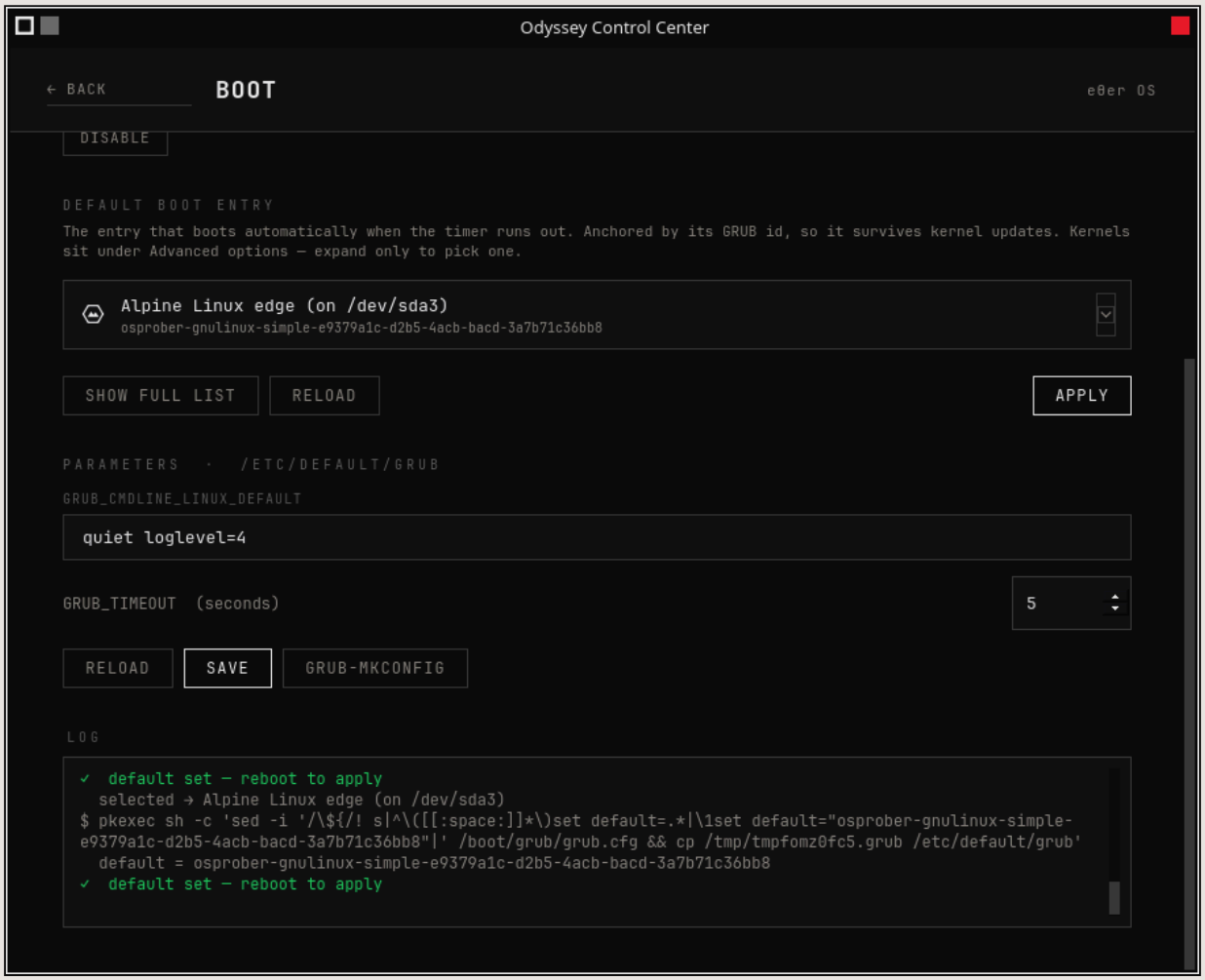

FIG · Nach APPLY: das Protokoll bestätigt „default set — reboot to apply“, und der Wähler zeigt jetzt Alpine Linux als aktiven Standard.

WHAT	Boot → GRUB-Tab → klicke auf den aktuellen Standardeintrag → wähle einen anderen aus dem Dropdown → klicke auf **APPLY**.
WHY	Du nutzt Dual-Boot und möchtest, dass ein anderes System automatisch startet, oder ein Kernel-Update hat einen alten Kernel ausgewählt zurückgelassen und du willst zum neuesten zurück.
RESULT	Der gewählte Eintrag startet ab dem nächsten Neustart automatisch. Die Änderung ist an die stabile ID dieses Eintrags gebunden, überlebt also künftige Kernel-Updates.
RISK	Keins für deine Daten — im schlimmsten Fall wählst du den falschen Eintrag und änderst ihn wieder.

SHOW FULL LIST zeigt jeden Eintrag genau so, wie GRUB selbst sie auflistet — nur lesbar, gruppiert, sodass Varianten pro einzelnem Kernel unter einer ausklappbaren Zeile „Advanced options“ liegen, statt die Hauptliste zu überladen. **RELOAD** liest die GRUB-Konfiguration neu, falls du vermutest, dass sie sich außerhalb des Panels geändert hat.

2.3 Visuelle Anleitung — der EFI-Tab

Klicke oben auf den Tab **EFI**. Das ist die Firmware-Ebene — was das Boot-Menü deines eigenen Mainboards zeigt, bevor überhaupt ein Betriebssystem gestartet ist.

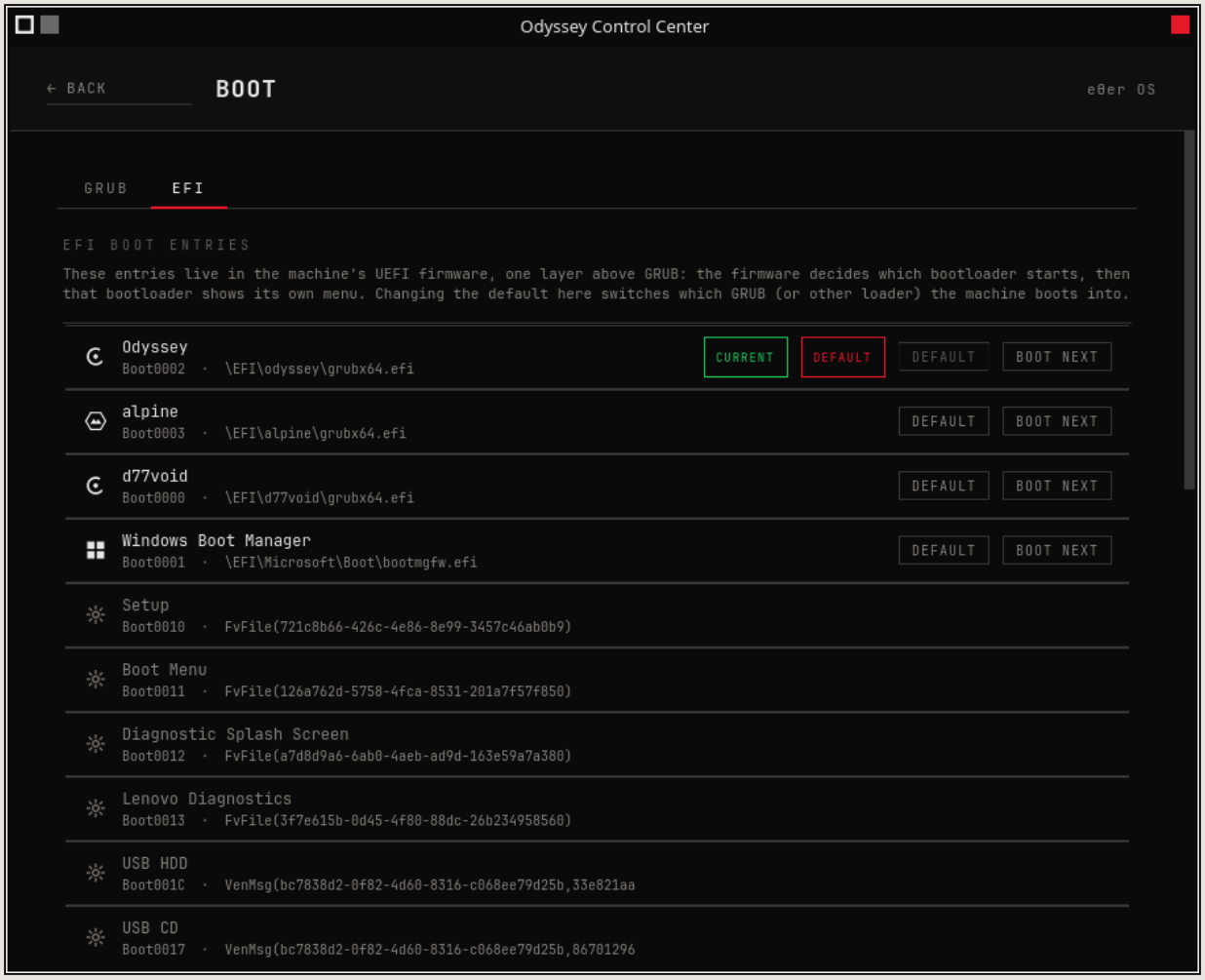

FIG · Der EFI-Tab: Odyssey markiert als CURRENT und DEFAULT in Grün/Rot, alpine, d77void und Windows Boot Manager jeweils mit eigenen DEFAULT-/BOOT-NEXT-Schaltflächen, plus die Firmware-Dienstprogramm-Einträge darunter.

Jede Zeile ist ein Firmware-Eintrag — sein Name, seine GRUB-ID (`Boot0002` und so weiter), und die genaue `.efi`-Datei, auf die er zeigt. Zwei Schaltflächen pro Zeile:

SCHALTFLÄCHE	WAS SIE TUT
DEFAULT	Macht diesen Eintrag zum dauerhaften Standard der Firmware — er startet ab jetzt bei jedem Neustart.
BOOT NEXT	Eine einmalige Ausnahme — dieser Eintrag startet nur beim **nächsten** Neustart, danach setzt sich die normale Reihenfolge automatisch fort. Ein Banner erscheint oben, solange eine Ausnahme aussteht, mit einer CANCEL-Schaltfläche.

WHAT	Boot → EFI-Tab → finde den gewünschten Eintrag (z. B. „Odyssey“ oder „Windows Boot Manager“) → klicke auf DEFAULT, oder BOOT NEXT für eine einmalige Ausnahme.
WHY	Wenn das Boot-Menü deines Mainboards komplett den falschen Bootloader startet — keine GRUB-Einstellung, eine Firmware-Einstellung — ist das der einzige Ort, der das behebt.
RESULT	DEFAULT: der Eintrag rückt dauerhaft an die Spitze der Firmware-Bootreihenfolge. BOOT NEXT: er startet einmal, dann wird zurückgesetzt.
RISK	Gering, und immer von diesem Tab oder vom UEFI-Setup-Bildschirm deines eigenen Mainboards (meist Entf oder F2 beim Einschalten) wiederherstellbar, falls etwas unerwartet startet.

2.4 Kernel-Parameter

Der letzte Abschnitt des GRUB-Tabs ist die reine Kernel-Befehlszeile — bearbeitbar wie ein normales Textfeld, keine GRUB-Syntax erforderlich — und das Menü-Timeout in Sekunden. **SAVE** schreibt sie in `/etc/default/grub`; **GRUB-MKCONFIG** erzeugt das eigentliche Boot-Menü aus dieser Datei neu, der Schritt, der einen gespeicherten Parameter beim nächsten Start wirksam macht. Das Panel wendet beim Neuerzeugen immer deine aktuelle Standardwahl erneut an, sodass dies nie zurücksetzt, welches System standardmäßig startet.

```
▸ ADVANCED

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

2.5 Advanced

WAS TATSÄCHLICH PASSIERT

Odysseys GRUB-Konfiguration ist **flach** — keine `submenu`-Blöcke. Jeder Kernel ist ein `menuentry` der obersten Ebene, was bedeutet, dass eine stabile ID (keine numerische Position) ihn immer korrekt auswählen kann, selbst über Kernel-Updates hinweg. Die „Advanced options“-Gruppierung, die du im Wähler siehst, ist reine visuelle Bequemlichkeit, gebaut durch Abgleich der gemeinsamen UUID zwischen dem einfachen Eintrag eines Betriebssystems und seinen Kernel-Varianten — die GRUB-Datei selbst hat diese Gruppierung nicht.

Einen Standard festzulegen schreibt in derselben Aktion an zwei Stellen: das persistente `GRUB_DEFAULT` in `/etc/default/grub`, und — weil `grub.cfg` möglicherweise nicht sofort neu erzeugt wird — eine direkte, chirurgische `sed`-Bearbeitung der aktiven Zeile `set default="..."` in `/boot/grub/grub.cfg` selbst, unter Umgehung des One-Shot-Zweigs `${next_entry}`, den `grub-reboot` verwendet. Deshalb wirkt ein Standard beim nächsten Start, ohne zuvor ein vollständiges `grub-mkconfig` zu erzwingen.

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

OS-PROBER VON HAND

```
$ sudo xbps-install -y os-prober
# Scan aktivieren — bearbeitet GRUB_DISABLE_OS_PROBER in /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 VON HAND — EFIBOOTMGR

Der EFI-Tab ist eine Oberfläche für `efibootmgr`. Alles, was er tut, entspricht einem Befehl:

```
# listet alle Firmware-Einträge, in der aktuellen Bootreihenfolge
$ sudo efibootmgr -v
# ordnet neu: Eintrag 0002 zuerst (= DEFAULT)
$ sudo efibootmgr -o 0002,0000,0001,0003
# einmaliger Start in Eintrag 0003 (= BOOT NEXT)
$ sudo efibootmgr -n 0003
# einen ausstehenden BootNext abbrechen
$ sudo efibootmgr -N
```

VERWAISTE LOADER

Gelegentlich schreibt ein Installer seinen Bootloader auf die EFI System Partition, ohne einen Firmware-Eintrag dafür zu registrieren — der Loader existiert auf der Festplatte, aber die Firmware weiß nicht, dass er da ist. Odysseys Panel scannt die ESP direkt und listet diese getrennt als **„ON DISK, NOT IN FIRMWARE“** auf, jeweils mit einem REGISTER per Klick. Von Hand ist das Äquivalent `efibootmgr -c -d /dev/sdX -p N -L "Name" -l '\EFI\name\bootx64.efi'` — beachte, dass neu erstellte Einträge von der Firmware selbst **zuerst** in die Bootreihenfolge gesetzt werden, wende also danach DEFAULT erneut auf deinen gewohnten Eintrag an, falls er nicht die Führung übernehmen soll.

DATEIEN, DIE DIESES PANEL BERÜHRT

DATEI	ROLLE
`/etc/default/grub`	Kernel-Befehlszeile, Timeout, GRUB_DEFAULT, GRUB_DISABLE_OS_PROBER
`/boot/grub/grub.cfg`	Das eigentliche erzeugte Boot-Menü — niemals von Hand bearbeiten; wird von grub-mkconfig überschrieben
EFI-Variablen (NVRAM)	Firmware-Booteinträge — überhaupt keine Datei; wird ausschließlich über efibootmgr gelesen und geschrieben

CHAPTER 3

Kernel

Vier Tabs — **SYSCTL**, **CPU GOVERNOR**, **HUGEPAGES**, **MODULES** — plus eine Reihe von Ein-Klick-Voreinstellungen über allen vieren, die auf alle gleichzeitig wirkt.

3.1 Visuelle Anleitung — die Voreinstellungen

Oben: die Version deines laufenden Kernels, der aktive Scheduler, und vier Voreinstellungs-Schaltflächen — **DESKTOP**, **GAMING**, **SERVER**, **BALANCED**.

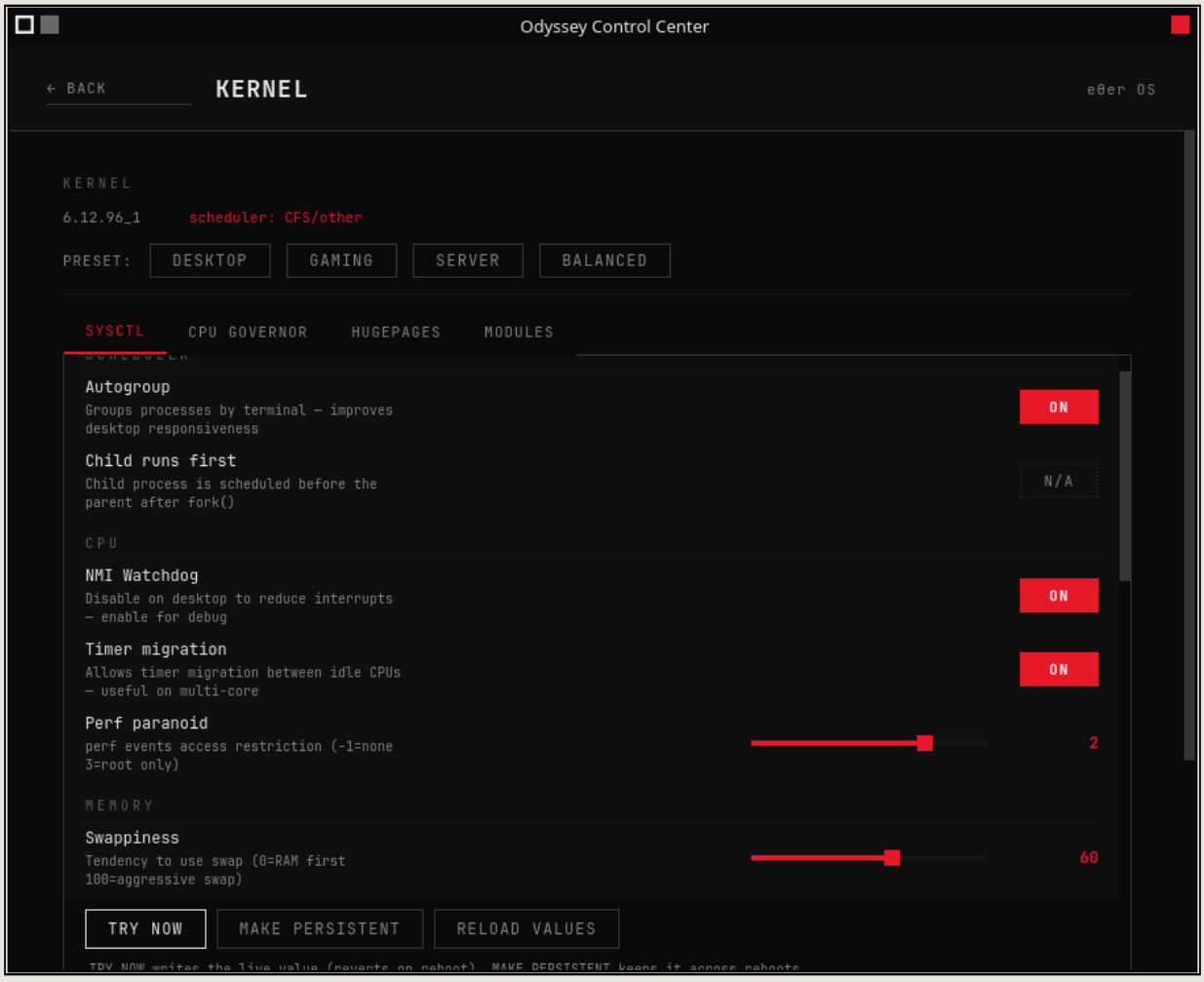

FIG · Der frisch geöffnete SYSCTL-Tab: Kernel 6.12.96_1, Scheduler CFS/other, die vier Voreinstellungen, und die Gruppen Scheduler/CPU/Memory darunter.

VOREINSTELLUNG	WANN SIE SICH LOHNT
DESKTOP	Eine Alltagsmaschine — Surfen, Büroarbeit, leichtes Multitasking. Niedrigere Swappiness, moderater Cache-Druck, Timer-Migration aktiv für sanften Energieverbrauch im Leerlauf. Der sinnvolle Standard für die meisten Leute.
GAMING	Swappiness fast auf null gedrückt (alles im RAM behalten), Timer-Migration deaktiviert (Kerne bleiben zugeteilt, statt Arbeit zu migrieren), Perf-Event-Einschränkungen für Profiling-Tools wie MangoHud gelockert. Nutze es, wenn Frame-Zeiten wichtiger sind als der Energieverbrauch im Leerlauf.
SERVER	Höhere Swappiness und Toleranz gegenüber Cache-Druck, NMI-Watchdog aktiv, strengere Einschränkungen bei Kernel-Zeigern/Protokollen. Nutze es auf einer kopflosen Maschine, die Stabilität und Sicherheit über interaktive Reaktionsfähigkeit stellt.
BALANCED	Ein Mittelweg zwischen Desktop und Gaming — für eine Maschine, die mal Arbeitsstation, mal Gaming-Sitzung ist, und du willst nicht ständig zwischen Voreinstellungen wechseln.

WHAT	Kernel → klicke auf DESKTOP, GAMING, SERVER oder BALANCED.
WHY	Ein Dutzend sysctl-Werte von Hand anzupassen ist viel verlangt von jemandem, der einfach nur „gute Einstellungen zum Spielen“ will — eine Voreinstellung ist genau das, bereits entschieden.
RESULT	Jeder Wert im SYSCTL-Tab springt auf die Zahlen der Voreinstellung, bereit und sichtbar — du brauchst trotzdem TRY NOW oder MAKE PERSISTENT (unten), um sie wirklich anzuwenden.
RISK	Keins — eine Voreinstellung zu wählen füllt nur das Formular.

3.2 Visuelle Anleitung — SYSCTL, Gruppe für Gruppe

Scrolle nach unten und jeder Wert ist unter einer Überschrift gruppiert, jeweils mit einer Beschreibung in einfacher Sprache direkt im Panel — du musst nie wissen, was `vm.compaction_proactiveness` bedeutet, ohne es zuerst dort zu lesen.

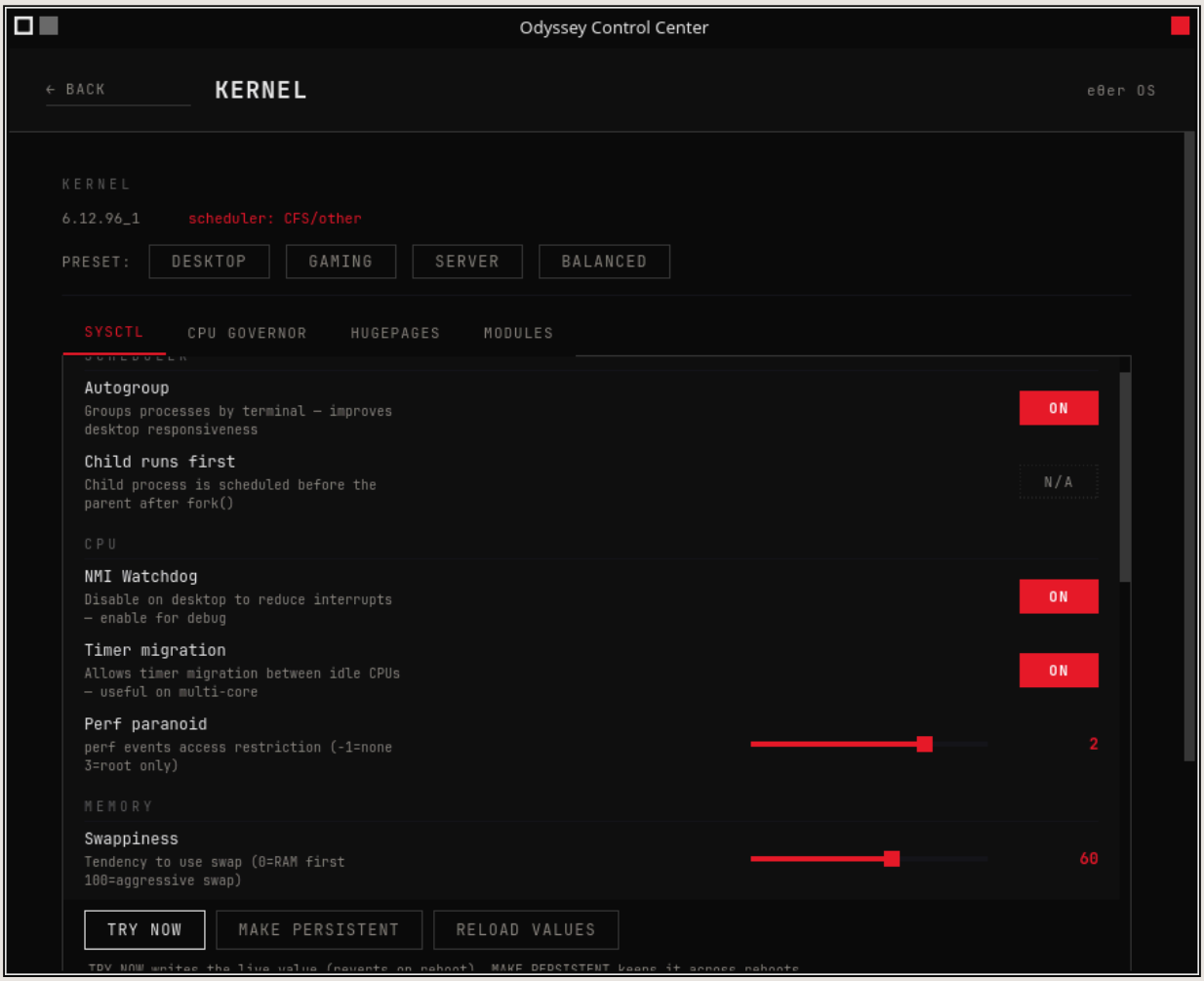

FIG · Gruppen Scheduler und CPU: Autogroup, Child runs first, NMI Watchdog, Timer migration, Perf paranoid — jeweils mit eigenem Schalter oder Regler und einer Erklärungszeile.

GRUPPE	WAS SIE ENTHÄLT
Scheduler	Wie Prozesse gruppiert und priorisiert werden
CPU	Watchdog, Timer-Migration, Einschränkungen bei der Leistungsüberwachung
Memory	Swappiness, Cache-Druck, Schwellenwerte für das Zurückschreiben schmutziger Seiten
Network	TCP Fast Open, Größe der Backlog-Warteschlange, Grenzwerte der Socket-Puffer
Security	Ob Kernel-Zeiger und Protokollmeldungen vor Benutzern ohne Rechte verborgen werden

Zwei Schaltflächen unten wenden deine Änderungen an — und sie sind absichtlich unterschiedlich:

WHAT	TRY NOW
WHY	Einen Wert ohne jedes Risiko testen, dass er einen missglückten Neustart übersteht.
RESULT	Live über `sysctl` geschrieben — sofort aktiv, nur für diese Sitzung.
RISK	Keins — setzt beim Neustart automatisch zurück.

WHAT	MAKE PERSISTENT
WHY	Du hast einen Wert (oder eine ganze Voreinstellung) getestet und willst, dass er Neustarts übersteht.
RESULT	In `/etc/sysctl.d/99-odyssey.conf` geschrieben **und** im selben Klick live angewendet.
RISK	Gering — es ist eine Textdatei; lösche die Zeile oder wende erneut an, um rückgängig zu machen.

RELOAD VALUES liest neu ein, was gerade aktiv ist, und verwirft alles Eingegebene, aber nicht Angewendete.

3.3 Visuelle Anleitung — CPU governor

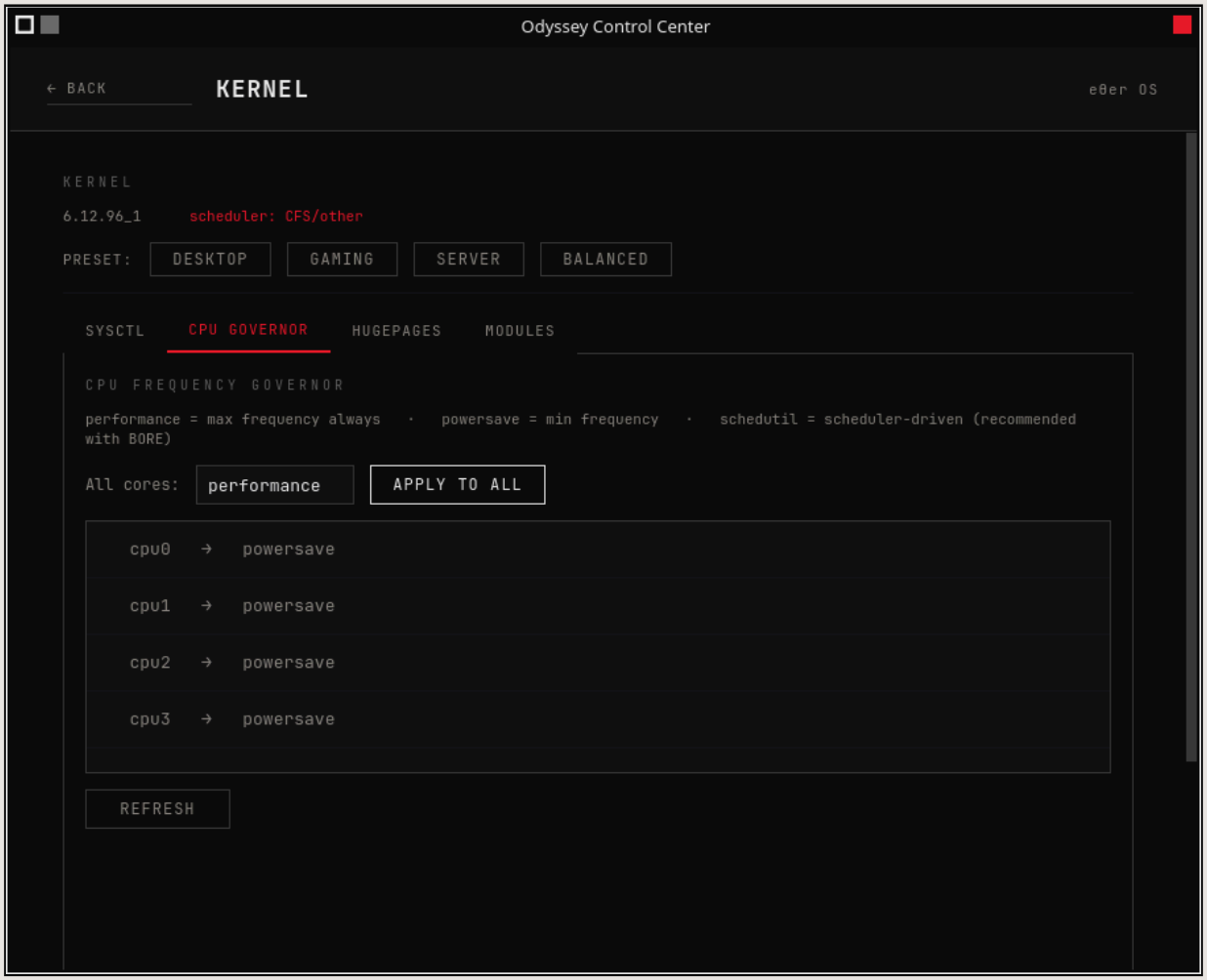

FIG · Der CPU-GOVERNOR-Tab: „All cores“ auf performance gesetzt mit APPLY TO ALL, und eine Liste pro Kern darunter, die noch alles auf powersave zeigt — eine Diskrepanz, die auf Korrektur wartet.

Der Governor entscheidet, wie aggressiv deine CPU die Taktfrequenz ändert. Ein Dropdown wendet einen Governor auf alle Kerne gleichzeitig an; die Liste darunter zeigt den **live** Governor pro einzelnem Kern — so entdeckst du einen Kern, der auf der falschen Einstellung hängt, genau wie im obigen Screenshot, wo „All cores“ performance sagt, aber jeder Kern darunter noch powersave zeigt, was bedeutet, dass APPLY TO ALL noch nicht geklickt wurde.

GOVERNOR	VERHALTEN
performance	Maximale Frequenz, immer — am reaktionsschnellsten, meister Verbrauch
powersave	Minimale Frequenz, immer — am leisesten, am wenigsten reaktionsschnell
schedutil	Frequenz gesteuert durch die eigene Lastsicht des Schedulers — empfohlen mit dem BORE-Scheduler

Nur die Governors, die deine spezifische CPU und dein Kernel tatsächlich anbieten, erscheinen im Dropdown.

3.4 Visuelle Anleitung — hugepages

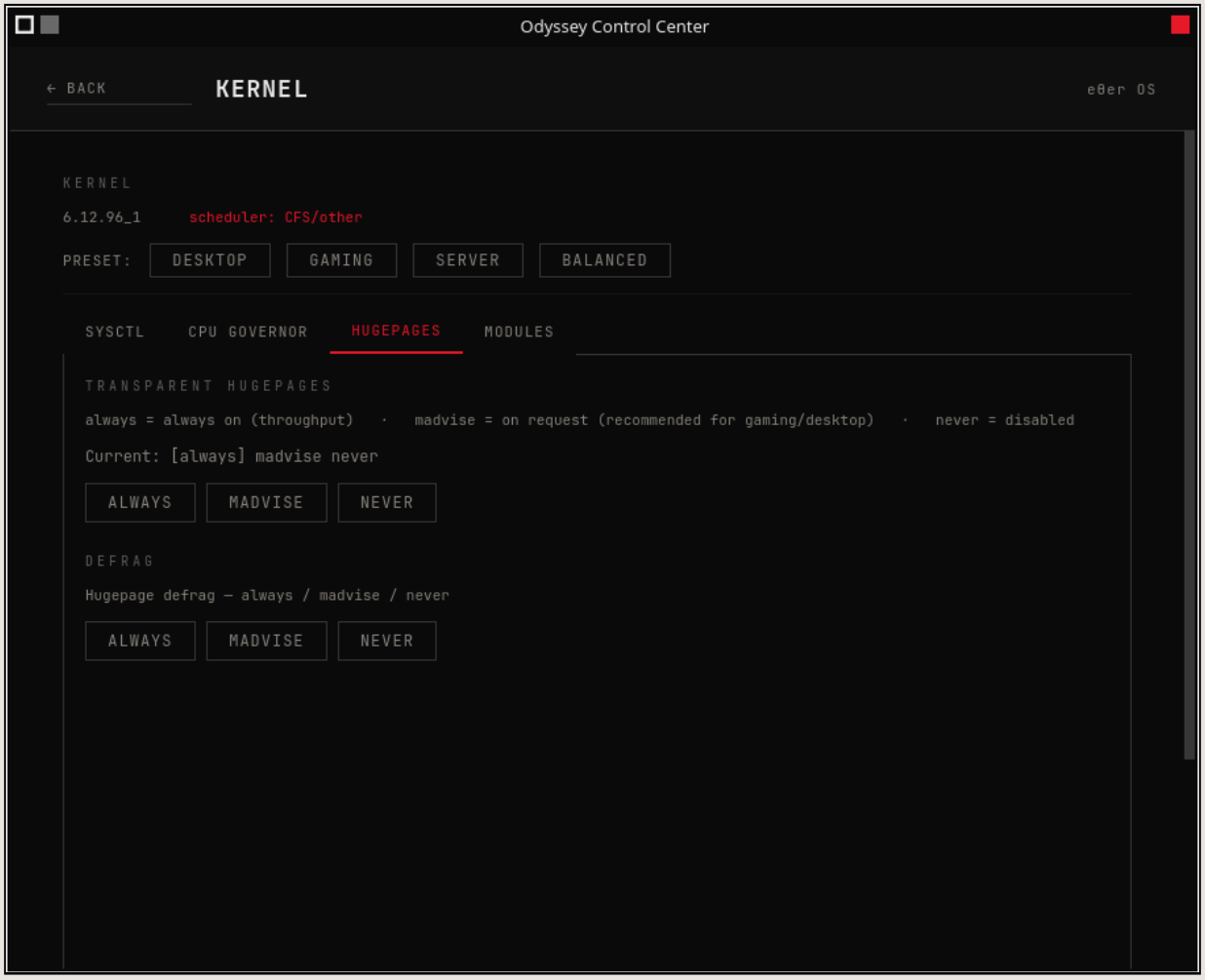

FIG · Der HUGEPAGES-Tab: Transparent Hugepages zeigt „Current: [always] madvise never“ mit hervorgehobenem ALWAYS, und darunter dieselbe Reihe von drei Schaltflächen erneut für Defrag.

Zwei unabhängige Einstellungen, jede eine Reihe von drei Schaltflächen, die sofort beim Klick angewendet werden — der aktuelle Live-Wert wird immer in eckigen Klammern über den Schaltflächen angezeigt, wie `[always] madvise never` im Screenshot.

EINSTELLUNG	WAS SIE STEUERT
Transparent hugepages	ALWAYS = überall immer aktiv (maximaler Durchsatz); MADVISE = nur wenn eine App es explizit anfordert (empfohlen für Gaming/Desktop); NEVER = vollständig deaktiviert
Defrag	Wie aggressiv der Kernel Speicher komprimiert, um Hugepages auf Anfrage zu **bilden** — dieselben drei Optionen, unabhängig von der obigen Einstellung

3.5 Visuelle Anleitung — modules

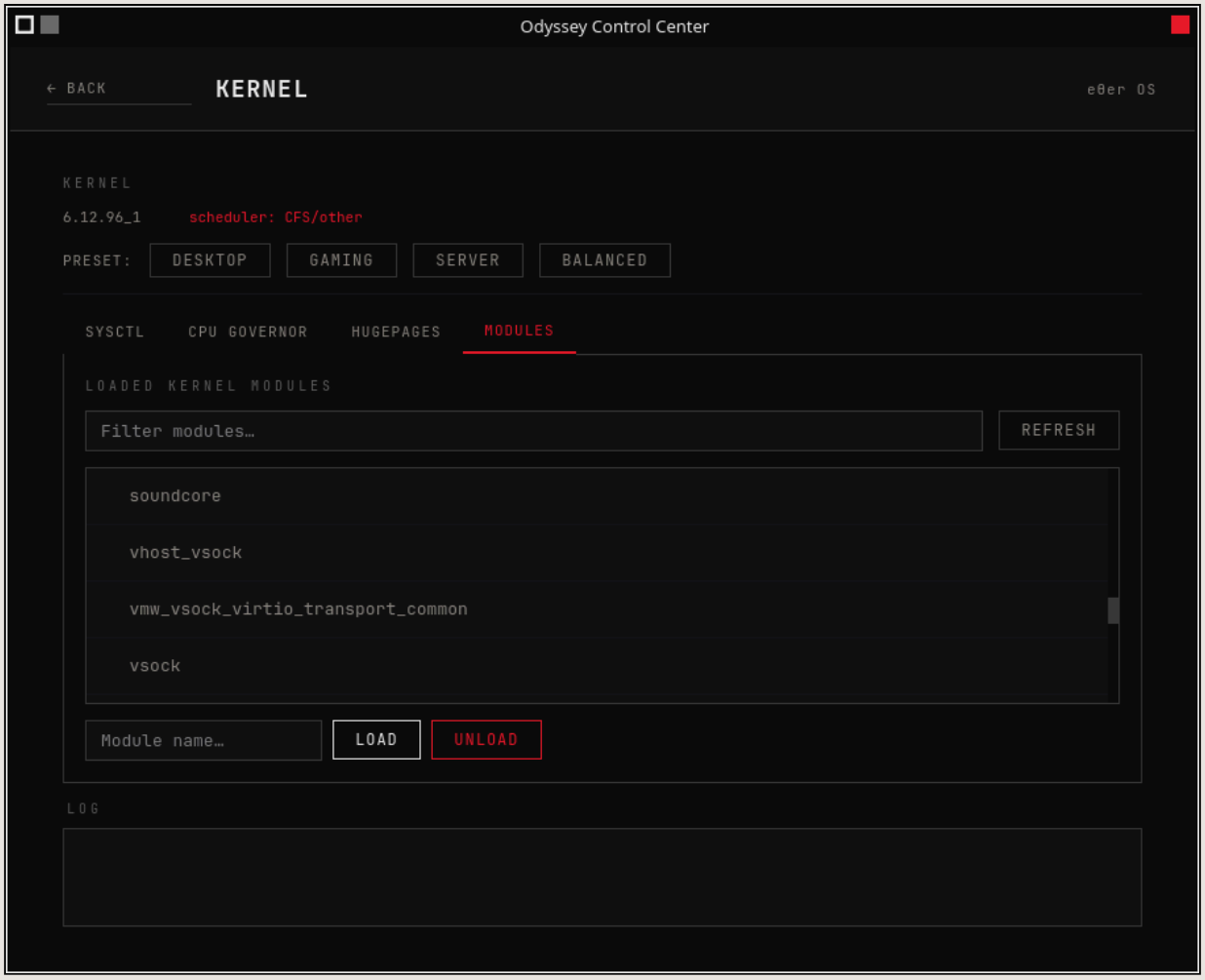

FIG · Der MODULES-Tab: ein Filterfeld, eine Live-Liste (soundcore, vhost_vsock, vsock…), und eine LOAD/UNLOAD-Zeile unten für ein nach Namen eingegebenes Modul.

Eine live gefilterte Liste jedes aktuell geladenen Kernel-Moduls — tippe in den Filter, um sie einzugrenzen, REFRESH zum Neulesen. Anders als bei den anderen drei Tabs kannst du hier direkt handeln: gib einen Modulnamen ein und klicke auf **LOAD** oder **UNLOAD**.

WHAT	Kernel → MODULES-Tab → gib einen Modulnamen ein → LOAD oder UNLOAD.
WHY	Gelegentlich brauchst du einen bestimmten Treiber oder Funktionsmodul aktiv (oder entfernt), ohne neu zu starten — Hardware testen, ein Gerät freigeben, das ein anderes Modul blockiert.
RESULT	Das Modul wird sofort in den laufenden Kernel eingefügt oder daraus entfernt.
RISK	Mäßig speziell für UNLOAD — ein Modul zu entfernen, von dem etwas anderes abhängt, kann destabilisieren, was es nutzte (ein Anzeigetreiber, ein Dateisystem). LOAD ist im Allgemeinen sicher; der Kernel weigert sich sofort, ein inkompatibles Modul zu laden, statt es halb anzuwenden.

▸ ADVANCED

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

3.6 Advanced

DIE EXAKTEN SYSCTL-SCHLÜSSEL

SCHLÜSSEL	BESCHRIFTUNG IM PANEL
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

VOREINSTELLUNGSWERTE, 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
# die vollständigen Voreinstellungstabellen lassen sich direkt in panel_kernel.py nach-
lesen —
# öffne den Quellcode des Control Center, wenn du jedes Feld willst
```

VON HAND, OHNE DAS PANEL

```
# live, temporär
$ sudo sysctl vm.swappiness=10
# persistent — dieselbe Datei, die das Panel schreibt
$ echo “vm.swappiness=10” | sudo tee -a /etc/sysctl.d/99-odyssey.conf
$ sudo sysctl –system
# CPU-Governor, pro Kern
$ 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
# module
$ sudo modprobe MODULE_NAME
$ sudo modprobe -r MODULE_NAME
```

WARUM TRY NOW UND MAKE PERSISTENT GETRENNTE AKTIONEN SIND

`sysctl`-Schreibvorgänge sind ihrer Natur nach immer nur live — der Kernel hat keine eingebaute Persistenz. MAKE PERSISTENT ist Odyssey, das eine einzige kanonische Datei wählt, `/etc/sysctl.d/99-odyssey.conf`, und sowohl das Live-Schreiben als auch das Datei-Schreiben zusammen ausführt, damit die beiden nie auseinanderlaufen. Diese Datei direkt zu bearbeiten und danach `sysctl --system` auszuführen, ist funktional identisch zum Klicken der Schaltfläche.

CHAPTER 4

Memory & Swap

Ein einziger Bildschirm, drei Abschnitte: eine nur lesbare Zusammenfassung, Swappiness, und ZRAM.

4.1 Visuelle Anleitung

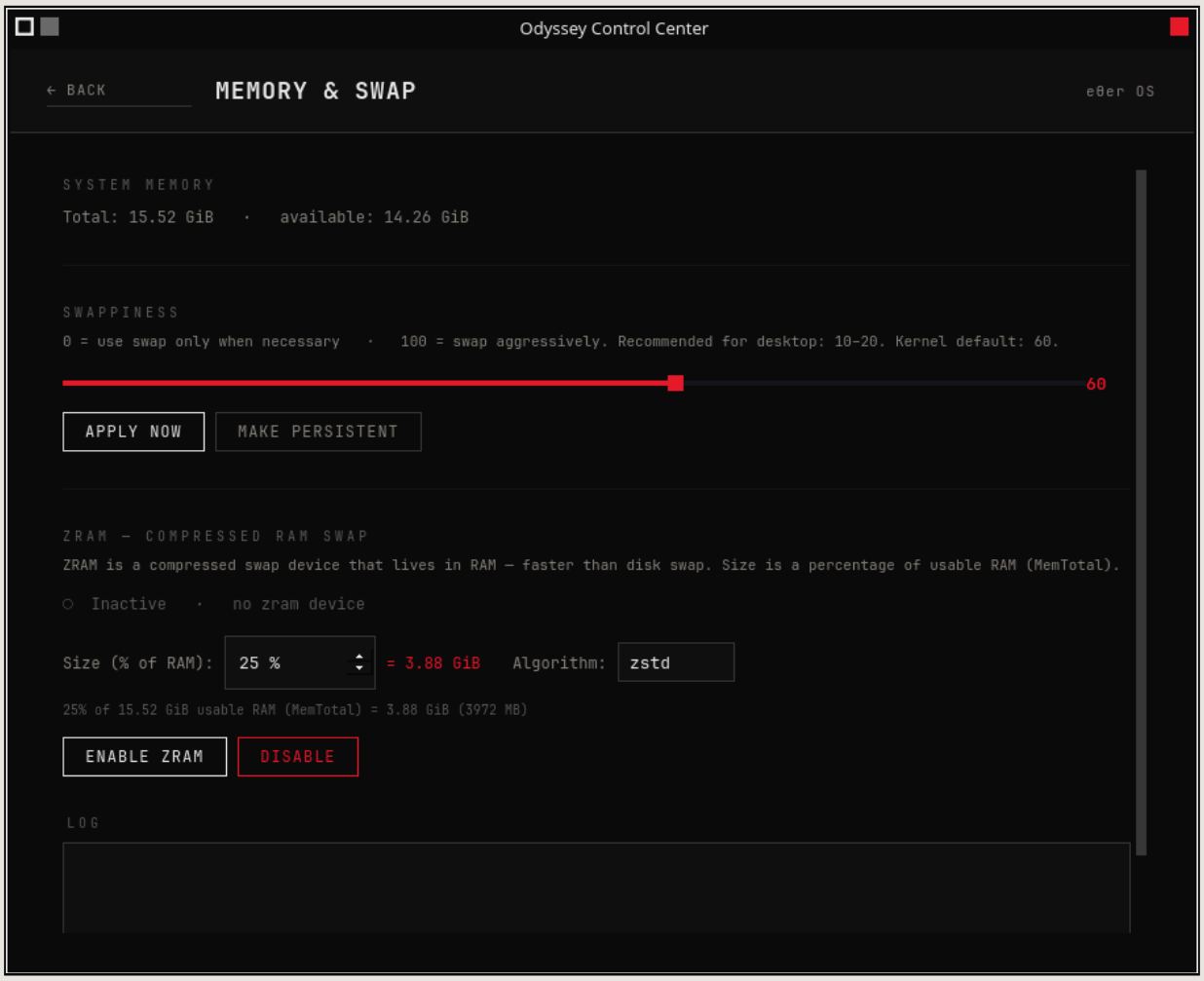

FIG · Das vollständige Panel: System memory (15,52 GiB gesamt, 14,26 GiB verfügbar), der Swappiness-Regler bei 60 mit APPLY NOW / MAKE PERSISTENT, und ZRAM darunter mit „Inactive · no zram device“, Größe, Algorithmus und ENABLE/DISABLE.

SYSTEM MEMORY

Nur eine Zusammenfassung oben — gesamter RAM und was gerade verfügbar ist. Keine Steuerelemente; es ist der Kontext für die beiden Abschnitte darunter.

SWAPPINESS

Ein einzelner Regler von 0 bis 100. 0 bedeutet „nur swappen, wenn es wirklich keine andere Wahl gibt“; 100 bedeutet „aggressiv swappen, lange bevor der RAM voll ist.“ Der Kernel-Standard ist 60 (was der Screenshot zeigt); der eigene Hinweis des Panels schlägt 10–20 für einen Desktop vor — dieselbe Zahl, die die DESKTOP-Voreinstellung des Kernel-Panels setzt.

WHAT	Ziehe den Regler → APPLY NOW zum Testen, MAKE PERSISTENT, sobald du zufrieden bist.
WHY	Ein niedrigerer Wert hält, was du aktiv nutzt, im schnellen RAM, statt es in den langsameren Swap zu verschieben — meist das, was du bei ausreichend installiertem Speicher willst.
RESULT	APPLY NOW ändert es sofort live, nur für diese Sitzung. MAKE PERSISTENT schreibt es in eine Konfigurationsdatei, damit es auch Neustarts übersteht.
RISK	Sehr gering. Ein extrem niedriger Wert auf einer wirklich knappen RAM-Maschine kann sie anfälliger dafür machen, unter starker Last plötzlich den Speicher auszugehen, statt sanft zu swappen — falls das passiert, stelle ihn wieder höher.

ZRAM — KOMPRIMIERTER SWAP IM RAM

ZRAM ist ein Swap-Gerät, das **innerhalb** des RAM selbst lebt, komprimiert — deutlich schneller als Swap auf Festplatte, weil keine Festplatte beteiligt ist. Die Größe ist ein Prozentsatz deines gesamten nutzbaren RAM, keine feste Zahl, sodass sich die Einstellung sinnvoll an unterschiedliche Maschinen anpasst; ein Live-Wert neben dem Größenfeld rechnet den Prozentsatz für dich in GiB um (im Screenshot: 25 % = 3,88 GiB auf einem System mit 15,52 GiB).

WHAT	Lege den Größenprozentsatz fest und wähle einen Algorithmus (zstd, lz4, lzo, oder deflate) → ENABLE ZRAM.
WHY	Komprimierter RAM-Swap absorbiert Speicherdruck mit deutlich geringerer Leistungseinbuße als Swap auf Festplatte — nützlich auf jeder Maschine, besonders bei begrenztem RAM.
RESULT	Ein `zram0`-Gerät wird erstellt und mit der gewählten Größe und dem gewählten Algorithmus aktiviert; die Statuszeile wechselt von „Inactive“ zu aktiv.
RISK	Gering. zstd (der Standard) balanciert für die meisten Maschinen Kompressionsverhältnis und CPU-Kosten gut aus. DISABLE deaktiviert es jederzeit sauber.

▸ ADVANCED

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

4.2 Advanced

WAS „HARDWAREABHÄNGIG“ HIER TATSÄCHLICH BEDEUTET

Beide Einstellungen interagieren mit deiner spezifischen Hardware, nicht nur mit „mehr RAM = besser“:

- **Schnelle NVMe-SSD**: eine Swappiness im Bereich 10–20 kostet sehr wenig, selbst wenn geswappt wird — die Platte ist schnell genug, dass es kaum auffällt. ZRAM gewinnt trotzdem bei Latenz und SSD-Verschleiß, aber der Abstand ist kleiner als bei einer mechanischen Festplatte.

- **Mechanische HDD oder eMMC als Swap**: halte die Swappiness niedrig (0–10) — Swap auf Festplatte auf langsamen Medien verursacht das Ruckeln, das man „Thrashing“ nennt. ZRAM ist genau hier ein viel größerer Gewinn.
- **CPU-Spielraum**: ZRAM tauscht Festplatten-I/O gegen CPU-Zyklen fürs Komprimieren und Dekomprimieren. Auf einer CPU-limitierten Maschine (ältere Hardware, oder eine, die beim Gaming bereits am Limit ist), komprimiert `lz4` schneller als `zstd` mit schlechterem Verhältnis — ein vernünftiger Kompromiss, wenn CPU-Zeit die knappere Ressource ist. Auf allem Modernen lohnt sich das bessere Verhältnis von `zstd` trotz etwas höherer Kosten.
- **RAM-Spielraum**: ein großer ZRAM-Prozentsatz (wie 50 %) auf einer Maschine mit nur 8 GiB gesamt lässt weniger echten RAM für Anwendungen übrig, da ZRAM selbst echten Speicher verbraucht, um die komprimierten Seiten zu speichern. Bei 16 GiB oder mehr sind 25 % ein komfortabler Standard mit Reserve.

VON HAND

```
# swappiness, live
$ sudo sysctl vm.swappiness=15
# swappiness, persistent — dieselbe Datei, die das Panel schreibt
$ echo “vm.swappiness=15” | sudo tee /etc/sysctl.d/99-odyssey.conf
# zram, manuelle einrichtung
$ 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
# überprüfen, dass es aktiv ist
$ swapon -show
```

WARUM DAS PANEL ZRAM0 ZURÜCKSETZT, STATT ES LIVE NEU ZU DIMENSIONIEREN

Die Größe eines ZRAM-Geräts kann nicht geändert werden, während es aktiv und als Swap in Benutzung ist — der Kernel verlangt, dass es zuerst deaktiviert und zurückgesetzt wird. Die Schaltfläche ENABLE ZRAM des Panels führt diese Abfolge automatisch aus (swapoff, reset, Neukonfiguration, swapon), weshalb das Ändern von Größe oder Algorithmus bei einem bereits aktiven Gerät trotzdem als eine einzige saubere Aktion erscheint statt als Fehler.

CHAPTER 5

Firewall

Das Panel Firewall hat sein eigenes, ausführliches Kapitel im **Security and Hardening Guide** — das Regelwerk zeilenweise erklärt, alle 21 Voreinstellungen mit ihren individuellen Risiken, der Editor für die Rohkonfiguration, und die Kommandozeilen-Äquivalente. Dieses Kapitel ist die visuell geführte Version.

5.1 Visuelle Anleitung

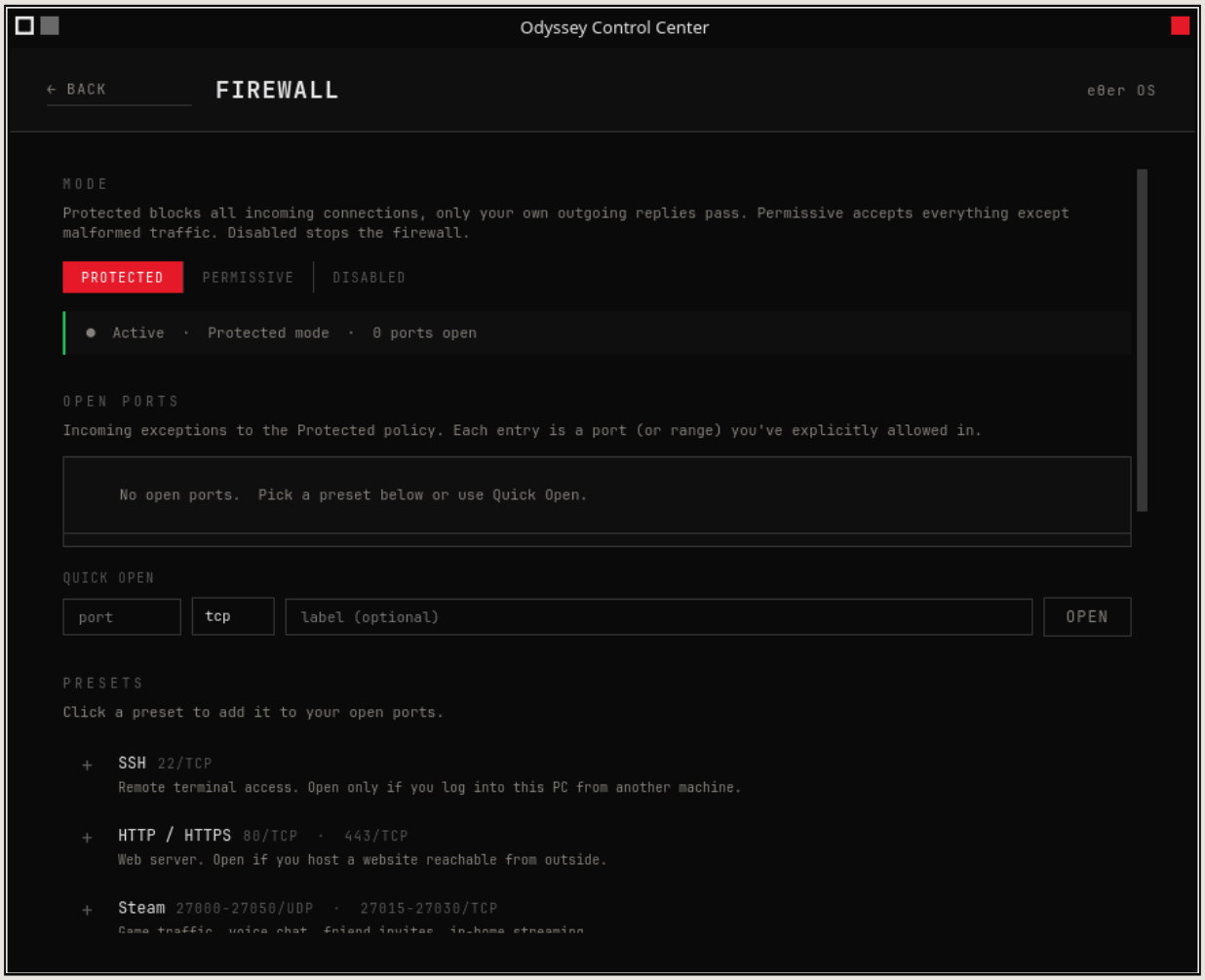

FIG · Modus PROTECTED aktiv, „0 ports open“, die Quick-Open-Felder, und die Voreinstellungskacheln beginnend mit SSH, HTTP/HTTPS und Steam.

MODE

Drei Schaltflächen oben, und das Panel erklärt jede in einer einzigen Zeile direkt darüber:

MODUS	WAS ER TUT
PROTECTED	Der Standard. Blockiert alle eingehenden Verbindungen; nur deine eigenen ausgehenden Antworten kommen durch.
PERMISSIVE	Akzeptiert alles außer fehlerhaftem Traffic — eine Diagnoseeinstellung für „ist die Firewall das, was das hier kaputt macht?“, kein Ort zum Bleiben.
DISABLED	Stoppt die Firewall vollständig.

Die Statuszeile darunter bestätigt, was tatsächlich aktiv ist — im Screenshot **„Active · Protected mode · 0 ports open.“**

OPEN PORTS UND VOREINSTELLUNGEN

OPEN PORTS listet deine expliziten Ausnahmen — standardmäßig leer, wie gezeigt. **QUICK OPEN** nimmt Port, Protokoll und optionale Beschriftung direkt entgegen, für alles, was nicht in der Voreinstellungsliste steht. **PRESETS** darunter bietet acht gängige Dienste mit je einem Klick, jeder mit einer Zeile, die erklärt, wann du ihn wirklich brauchst (SSH: „Fernzugriff auf das Terminal, öffne ihn nur, wenn du dich von einer anderen Maschine aus bei diesem PC anmeldest“), plus dreizehn weitere unter einem ausklappbaren „MORE PRESETS.“

WHAT	Firewall → klicke auf eine Voreinstellungskachel, oder fülle QUICK OPEN aus → SAVE & APPLY.
WHY	Du betreibst etwas — einen Spieleserver, einen Medienserver, einen Entwicklungsserver — das eine andere Maschine erreichen muss.
RESULT	Dieser Port öffnet sich; alles andere bleibt durch die PROTECTED-Richtlinie geschlossen.
RISK	Hängt vollständig von der Voreinstellung ab — sieh die Risikohinweise pro Voreinstellung im Security and Hardening Guide nach, bevor du etwas über SSH oder HTTP/HTTPS hinaus auf einer aus dem Internet erreichbaren Maschine öffnest.

Nichts wirkt sich aus, bis du auf **SAVE & APPLY** drückst — Änderungen werden zuerst zurückgestellt, sodass du genau überprüfen kannst, was du zu öffnen im Begriff bist, bevor du bestätigst.

```
▸ ADVANCED

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

5.2 Advanced

Die vollständige technische Referenz — das Standard-Regelwerk von `nftables.conf` regelweise erklärt, jede der 21 Voreinstellungen mit ihrer spezifischen Exposition und Abmilderung, Rezepte zur Beschränkung auf das reine LAN, die `nft`-Kommandozeilen-Äquivalente, und die Behebung einer ausgesperrten SSH-Sitzung — lebt in **Kapitel 2 des Security and Hardening Guide** (`/usr/share/doc/odyssey/` auf deinem installierten System). Dieses Handbuch verdoppelt es absichtlich nicht; betrachte jenes Kapitel als Fortsetzung dieses Abschnitts.

```
# der Befehl, den es sich jetzt schon zu kennen lohnt:
$ sudo nft list ruleset
# zeigt genau, was im Kernel geladen ist, unabhängig vom Panel
```

CHAPTER 6

AppArmor

Fünf Tabs: **STATUS**, **PROFILES**, **COMMUNITY**, **VIOLATIONS**, **EDITOR**. Derselbe Verweis wie bei Firewall: die vertiefte Referenz — was verpflichtende Zugriffskontrolle wirklich ist, der vollständige Confinement-Ablauf, ein praktisches Beispiel — lebt im Security and Hardening Guide. Hier ist die visuelle Anleitung.

6.1 Visuelle Anleitung — STATUS

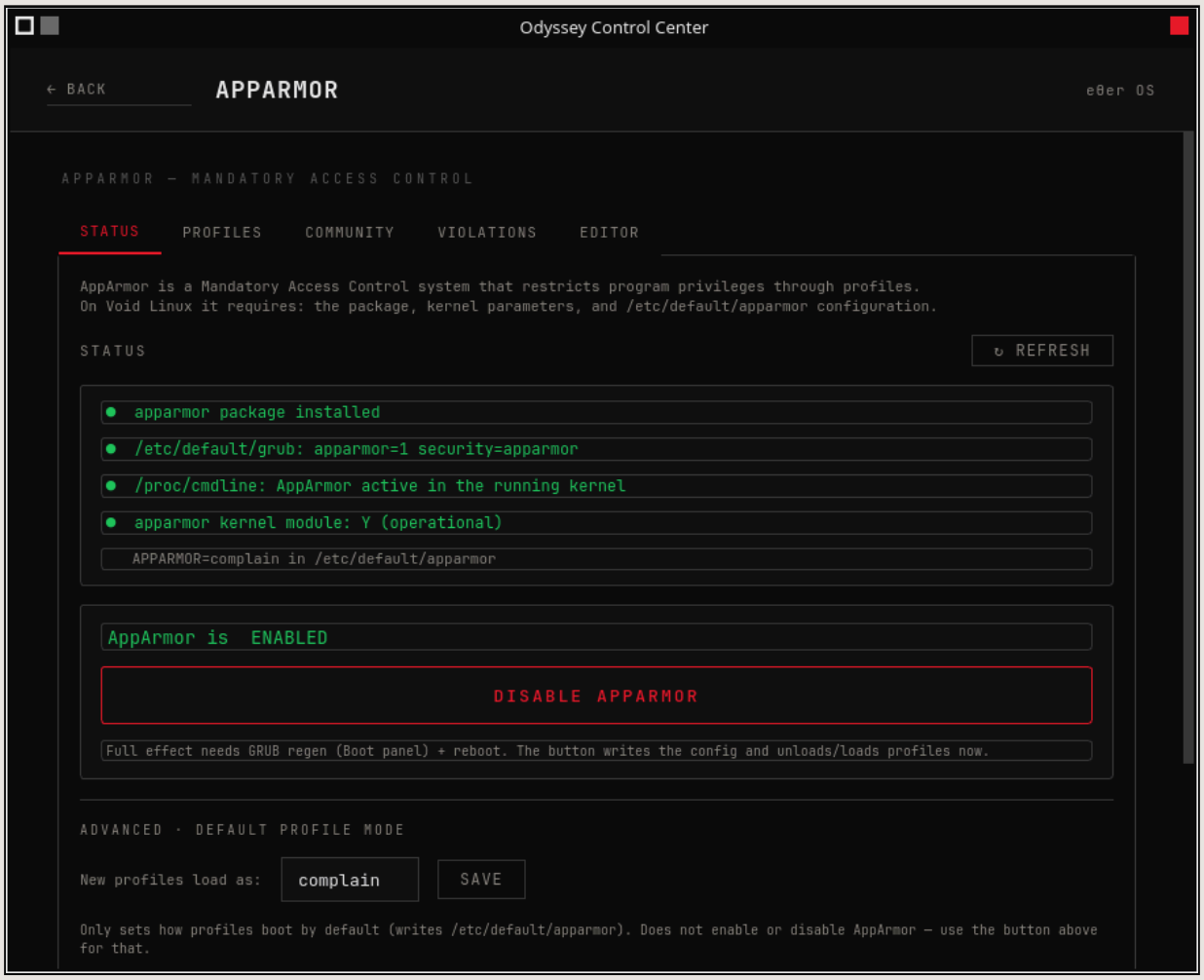

FIG · STATUS-Tab: vier grüne Häkchen (Paket installiert, GRUB-Parameter, laufender Kernel, Kernel-Modul), „AppArmor is ENABLED“ mit einer roten Schaltfläche DISABLE APPARMOR, und der Standard-Profilmodus auf complain gesetzt.

Vier Prüfungen verifizieren, dass das Framework tatsächlich funktioniert, von oben nach unten: das Paket `apparmor` ist installiert, die Kernel-Parameter in GRUB sind gesetzt, der **laufende** Kernel ist tatsächlich mit ihnen gestartet (das ist die, die hinterherhinken kann, wenn du nach dem Aktivieren nicht

neu gestartet hast), und das Kernel-Modul selbst ist betriebsbereit. Darunter: eine große Schaltfläche zum Aktivieren/Deaktivieren, und **ADVANCED · DEFAULT PROFILE MODE** — ob neu geladene Profile in complain oder enforce starten.

WHAT	AppArmor → STATUS → bestätige, dass alle vier Prüfungen grün sind.
WHY	Wenn eine Prüfung rot oder gelb ist, kann sonst nichts in diesem Panel korrekt funktionieren — das ist immer das Erste, was zu prüfen ist, wenn mit AppArmor etwas nicht zu stimmen scheint.
RESULT	Gewissheit, dass das Framework wirklich aktiv ist, nicht nur installiert.
RISK	Keins — nur lesende Überprüfung.

6.2 Visuelle Anleitung — PROFILES

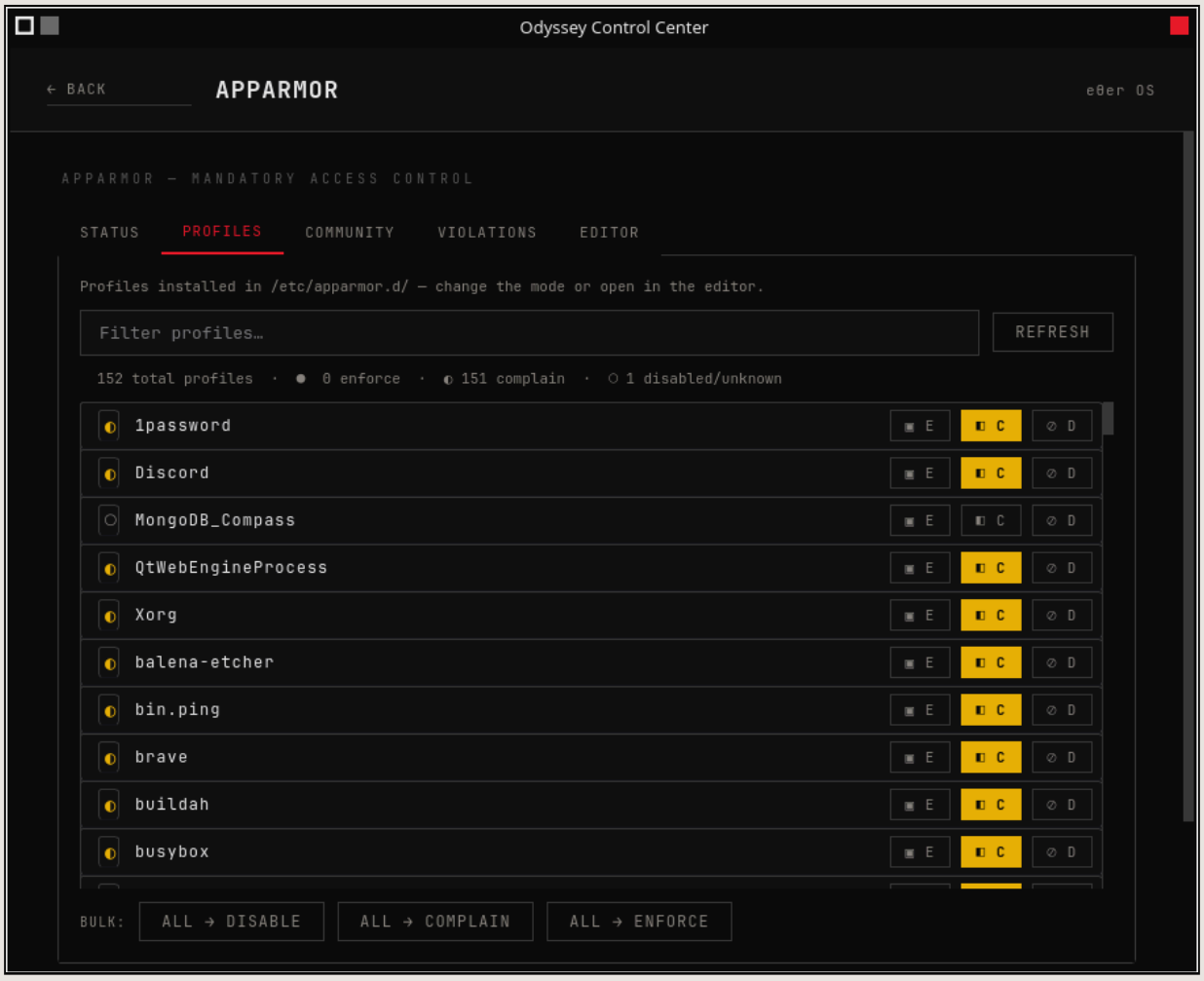

FIG · PROFILES-Tab: 152 Profile insgesamt, 0 enforce, 151 complain, 1 disabled/unknown — eine filterbare Liste mit 1password, Discord, MongoDB_Compass und mehr, jedes mit E-/C-/D-Schaltflächen, und unten ALL→DISABLE / ALL→COMPLAIN / ALL→ENFORCE.

Jedes geladene Profil, eine Zeile pro Profil, mit einem Filterfeld und einer Zählung oben (im Screenshot: 152 Profile, alle bis auf eins im complain-Modus — Odysseys Standard). Drei Schaltflächen pro Zeile:

SCHALTFLÄCHE	WIRKUNG
E	Schaltet dieses Profil auf enforce — Verstöße werden blockiert, nicht nur protokolliert
C	Schaltet auf complain — protokolliert Verstöße, blockiert nichts
D	Deaktiviert das Profil vollständig — das Programm läuft unbeschränkt

Die drei **BULK**-Schaltflächen unten wenden denselben Wechsel auf jedes Profil der Liste gleichzeitig an — nützlich direkt nach der Installation einer Charge aus COMMUNITY, aber behandle ALL→ENFORCE mit besonderer Vorsicht; siehe den Aktionsblock im nächsten Abschnitt.

6.3 Visuelle Anleitung — COMMUNITY

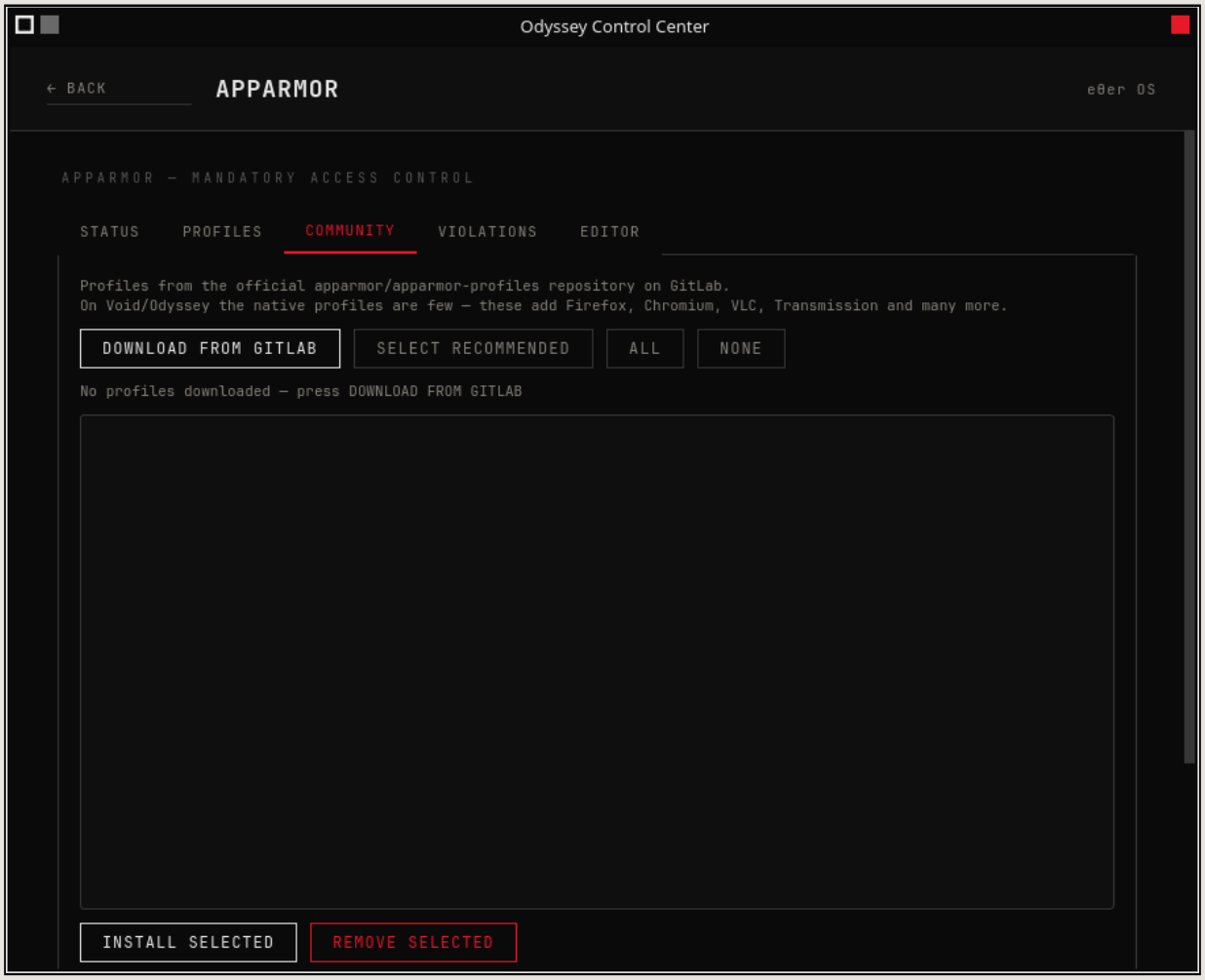

FIG · COMMUNITY-Tab: „No profiles downloaded — press DOWNLOAD FROM GITLAB“, mit den Schaltflächen DOWNLOAD FROM GITLAB, SELECT RECOMMENDED, ALL und NONE, und INSTALL SELECTED / REMOVE SELECTED unten.

AppArmor beschränkt ab Werk vor allem Systemkomponenten — die Desktop-Anwendungen, um die du dich wirklich sorgst (Browser, Chat-Apps, Medienplayer), brauchen Profile aus der Community. Dieser Tab holt sie direkt aus dem offiziellen Repository `apparmor/apparmor-profiles` auf GitLab.

1 **DOWNLOAD FROM GITLAB**

Holt die aktuelle Liste verfügbarer Profile — noch wird nichts installiert, das befüllt nur die Liste.

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

Wählt einen sinnvollen Desktop-Standardsatz — Browser der Firefox-Familie, Chromium, VLC, Transmission und Ähnliches.

3 **INSTALL SELECTED**

Installiert die ausgewählten Profile — sie laden im complain-Modus, wie alles andere, bereit für den Beobachten-dann-Anwenden-Zyklus unten.

WHAT	AppArmor → COMMUNITY → DOWNLOAD FROM GITLAB → SELECT RECOMMENDED (oder wähle einzeln) → INSTALL SELECTED.
WHY	Die Anwendungen, die am meisten nicht vertrauenswürdigen Eingaben ausgesetzt sind — ein Browser, der eine bösartige Seite rendert, ein Medienplayer, der eine manipulierte Datei öffnet — sind genau die, die ein Profil am besten schützt.
RESULT	Neue Profile erscheinen in PROFILES, im complain-Modus, protokollieren, was sie blockiert hätten, ohne bisher etwas zu blockieren.
RISK	Keins bei der Installation — nichts wird angewendet, bis du es in PROFILES entscheidest.

6.4 Visuelle Anleitung — VIOLATIONS

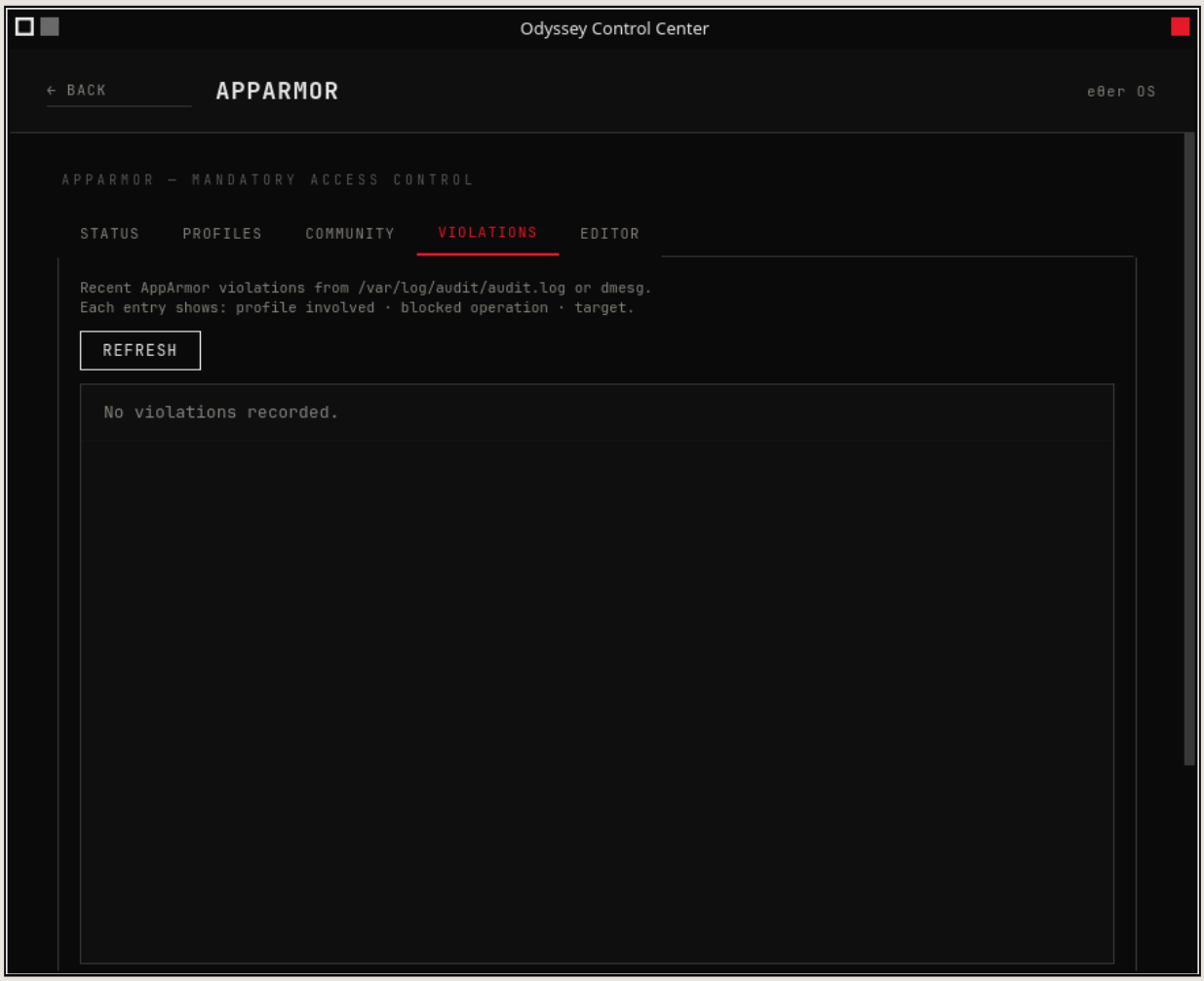

FIG · VIOLATIONS-Tab: „No violations recorded“, mit einer REFRESH-Schaltfläche und einem Hinweis, dass Einträge aus /var/log/audit/audit.log oder dmesg stammen.

Dieser Tab liest das AppArmor-Audit-Protokoll des Systems und zeigt, was ein Profil **blockiert hätte** (im complain-Modus) oder **blockiert hat** (im enforce-Modus) — jeder Eintrag nennt das Profil, die blockierte Operation, und die Zieldatei oder -ressource.

> **EINE LEERE LISTE IST DAS ERWARTETE BILD AUF EINEM GESUNDEN SYSTEM**
>
> Der Screenshot zeigt genau, was du typischerweise sehen wirst: keine Verstöße. Das ist kein Fehler noch ein Zeichen, dass das Panel nicht funktioniert — es bedeutet, dass jedes profilierte Programm seit der letzten Protokollprüfung innerhalb der erlaubten Grenzen geblieben ist. Verstöße erscheinen hier in dem Moment, in dem etwas Profiliertes versucht, etwas zu tun, das sein Profil nicht erlaubt; geh das betreffende Programm benutzen, komm zurück, und drücke REFRESH.

Sobald Einträge erscheinen, ist ein Doppelklick auf einen davon der schnelle Weg in EDITOR mit genau diesem bereits geöffneten Profil — siehe unten.

6.5 Visuelle Anleitung — EDITOR

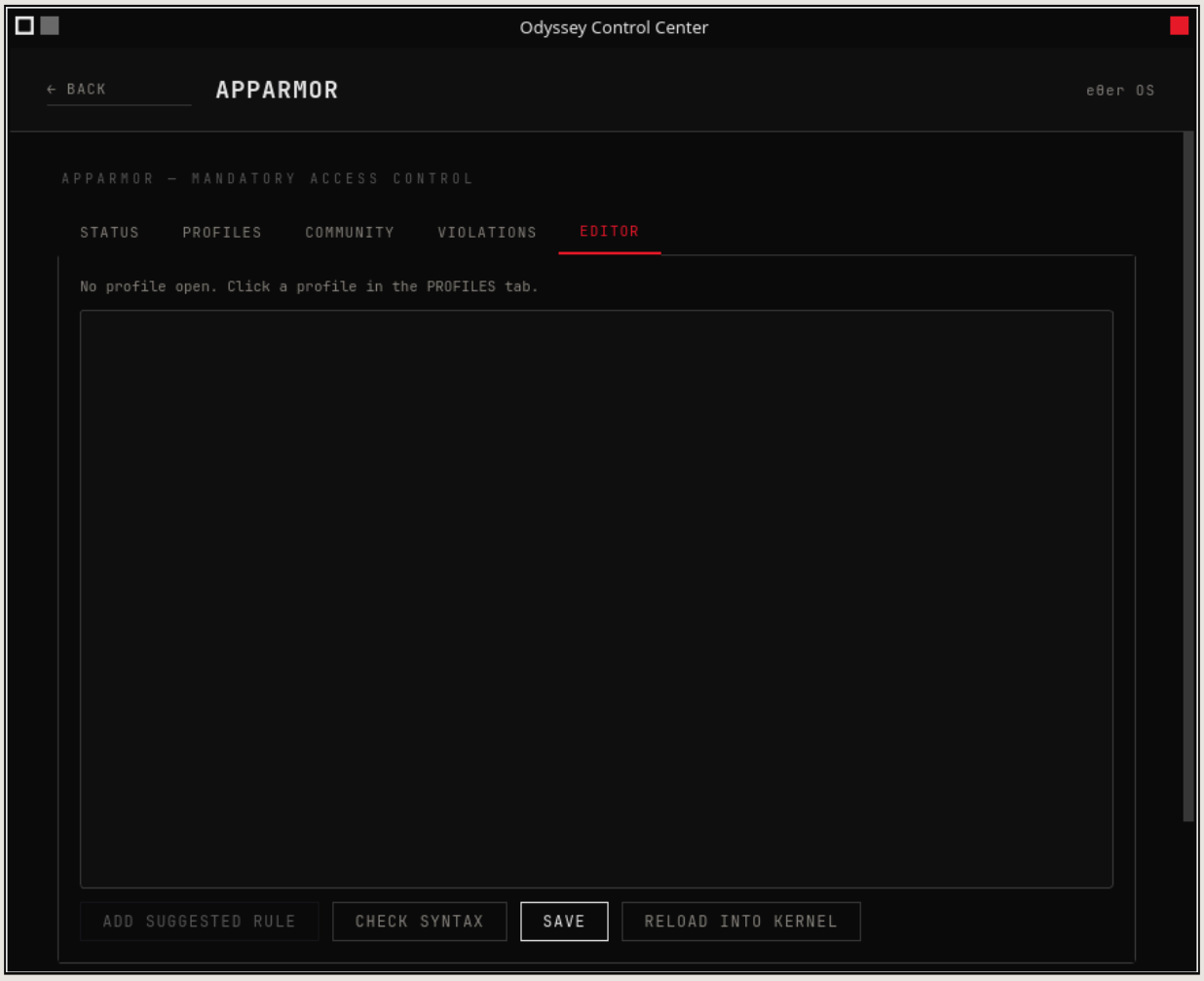

FIG · EDITOR-Tab: „No profile open. Click a profile in the PROFILES tab“, ein leeres Textfeld, und vier Schaltflächen — ADD SUGGESTED RULE, CHECK SYNTAX, SAVE, RELOAD INTO KERNEL — alle außer SAVE deaktiviert.

Der Editor öffnet sich absichtlich **leer**, wie gezeigt — er bearbeitet immer nur ein Profil gleichzeitig, und du wählst welches aus PROFILES (klicke auf eine Zeile) oder aus VIOLATIONS (Doppelklick auf einen Eintrag). Das verhindert den leichten Fehler, aus Versehen die falsche Datei zu bearbeiten.

SCHALTFLÄCHE	WAS SIE TUT
ADD SUGGESTED RULE	Nur aktiv, wenn du von einem VIOLATIONS-Eintrag kommst — fügt die exakte Regelzeile ein, die die protokollierte Aktion erlaubt hätte
CHECK SYNTAX	Validiert das Profil, ohne es zu laden — fängt einen Tippfehler ab, bevor er zählt
SAVE	Schreibt deine Änderungen in die Profildatei auf der Festplatte
RELOAD INTO KERNEL	Macht eine gespeicherte Änderung sofort wirksam, ohne auf den nächsten Start oder Dienst-Neustart zu warten

WHAT	PROFILES → klicke auf den Namen eines Profils → EDITOR öffnet es → nimm eine Änderung vor → CHECK SYNTAX → SAVE → RELOAD INTO KERNEL.
WHY	Das ist der komplette Confinement-Zyklus in einer Zeile: beobachte im complain-Modus (VIOLATIONS), korrigiere, was wirklich fehlt (EDITOR), erst dann schalte dieses Profil auf enforce (PROFILES).
RESULT	Das Profil verhält sich genau so wie bearbeitet, live, ohne Neustart.
RISK	Ein Profil falsch zu bearbeiten kann später unter enforce etwas Legitimes blockieren — CHECK SYNTAX fängt fehlerhafte Regeln vor SAVE ab, aber keine logischen Fehler. Wenn ein von dir bearbeitetes Profil unter enforce Probleme verursacht, setze es von PROFILES aus zurück auf complain und prüfe VIOLATIONS erneut.

▸ ADVANCED

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

6.6 Advanced

KOMMANDOZEILEN-ÄQUIVALENTE

```
$ sudo aa-status
# dieselben Zählungen wie die Kopfzeile des PROFILES-Tabs, aus dem Terminal
$ sudo aa-enforce /etc/apparmor.d/PROFILE_FILE
$ sudo aa-complain /etc/apparmor.d/PROFILE_FILE
$ sudo aa-disable /etc/apparmor.d/PROFILE_FILE
# äquivalent zu RELOAD INTO KERNEL nach einer manuellen Änderung
$ sudo apparmor_parser -r /etc/apparmor.d/PROFILE_FILE
```

DAS ROHE AUDIT-PROTOKOLL LESEN

```
$ sudo grep apparmor /var/log/audit/audit.log | tail -20
# oder, wenn kein dedizierter Audit-Daemon läuft:
$ sudo dmesg | grep -i apparmor
```

Nützliche Felder beim Lesen einer Zeile von Hand: `apparmor="DENIED"` (enforce) oder `apparmor="ALLOWED"` (complain, nur protokolliert), `operation=` (was versucht wurde), `profile=` (welches Profil entschieden hat — entspricht dem, was EDITOR öffnet), `name=` (der Pfad oder die Ressource), `comm=` (der Prozess).

WARUM ALL→ENFORCE IM BULK RISKANT IST

Ein Profil einzeln zu eskalieren, nachdem sein VIOLATIONS-Protokoll eine Weile still geblieben ist, ist das sichere Muster — du weißt, Profil für Profil, dass normale Nutzung es nicht auslöst. ALL→ENFORCE überspringt diese Beobachtungsphase für alle Profile gleichzeitig, einschließlich solcher, die du vielleicht gerade erst aus COMMUNITY installiert und nie wirklich benutzt hast. Ein Profil, das durchgesetzt wird, bevor es gelebt wurde, kann eine legitime Funktion genau in dem Moment blockieren, in dem du sie zum ersten Mal benutzt. Bevorzuge den Eins-nach-dem-anderen-Zyklus; hebe Bulk-Aktionen für ALL→COMPLAIN auf (immer sicher — lockert nur) oder ALL→DISABLE bei der Fehlersuche.

DATEIEN, DIE DIESES PANEL BERÜHRT

PFAD	ROLLE
`/etc/apparmor.d/`	Eine Profildatei pro beschränktem Programm — was PROFILES listet und EDITOR öffnet
`/etc/default/apparmor`	Standardmodus beim Start (`APPARMOR=complain`) und der enable/disable-Status
`/etc/default/grub`	Die Kernel-Parameter `apparmor=1 security=apparmor`, die STATUS prüft

CHAPTER 7

Privacy

Zwei unabhängige Werkzeuge, jedes mit seinem eigenen dreistufigen Ablauf install → enable → running, und eine Brücke zwischen beiden, sobald sie laufen.

7.1 Visuelle Anleitung — SearXNG, Schritt für Schritt

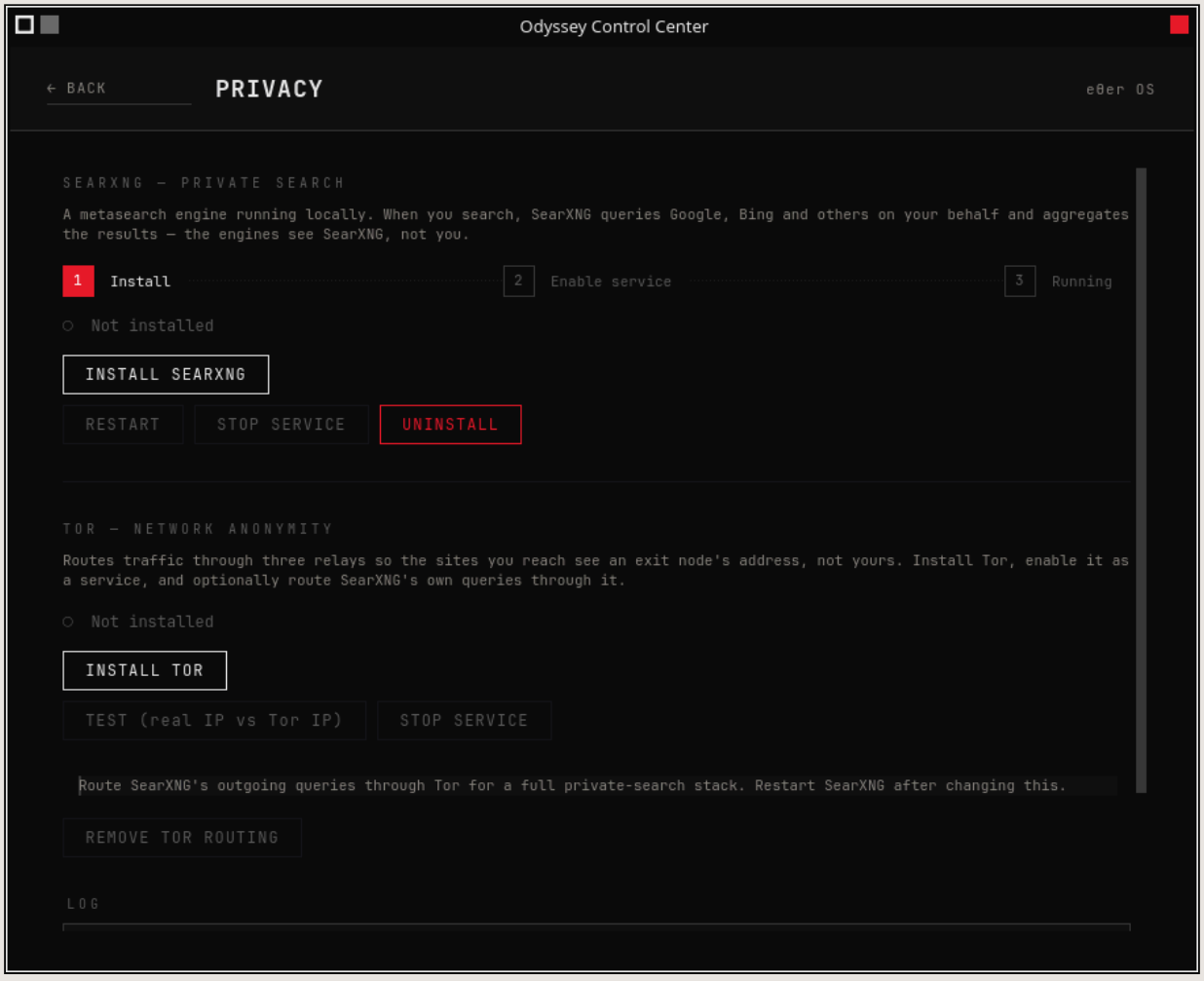

FIG · Das Panel im Ruhezustand: SearXNGs Anzeige bei Schritt 1 „Install“, Status „Not installed”, eine einzige Schaltfläche INSTALL SEARXNG; Tors Anzeige zeigt ebenfalls „Not installed“ mit INSTALL TOR darunter.

SearXNG ist eine Metasuchmaschine, die lokal auf deiner eigenen Maschine läuft — wenn du suchst, ist es SearXNG, das Google, Bing und die anderen in deinem Namen abfragt, sodass diese Suchmaschinen SearXNG die Anfrage stellen sehen, nie dich direkt.

1 **INSTALL SEARXNG**

Klicke. Das Protokoll zeigt die tatsächliche Installation im Gange — Klonen des Quellcodes, Erstellen eines Systembenutzers, Einrichten einer virtuellen Python-Umgebung.

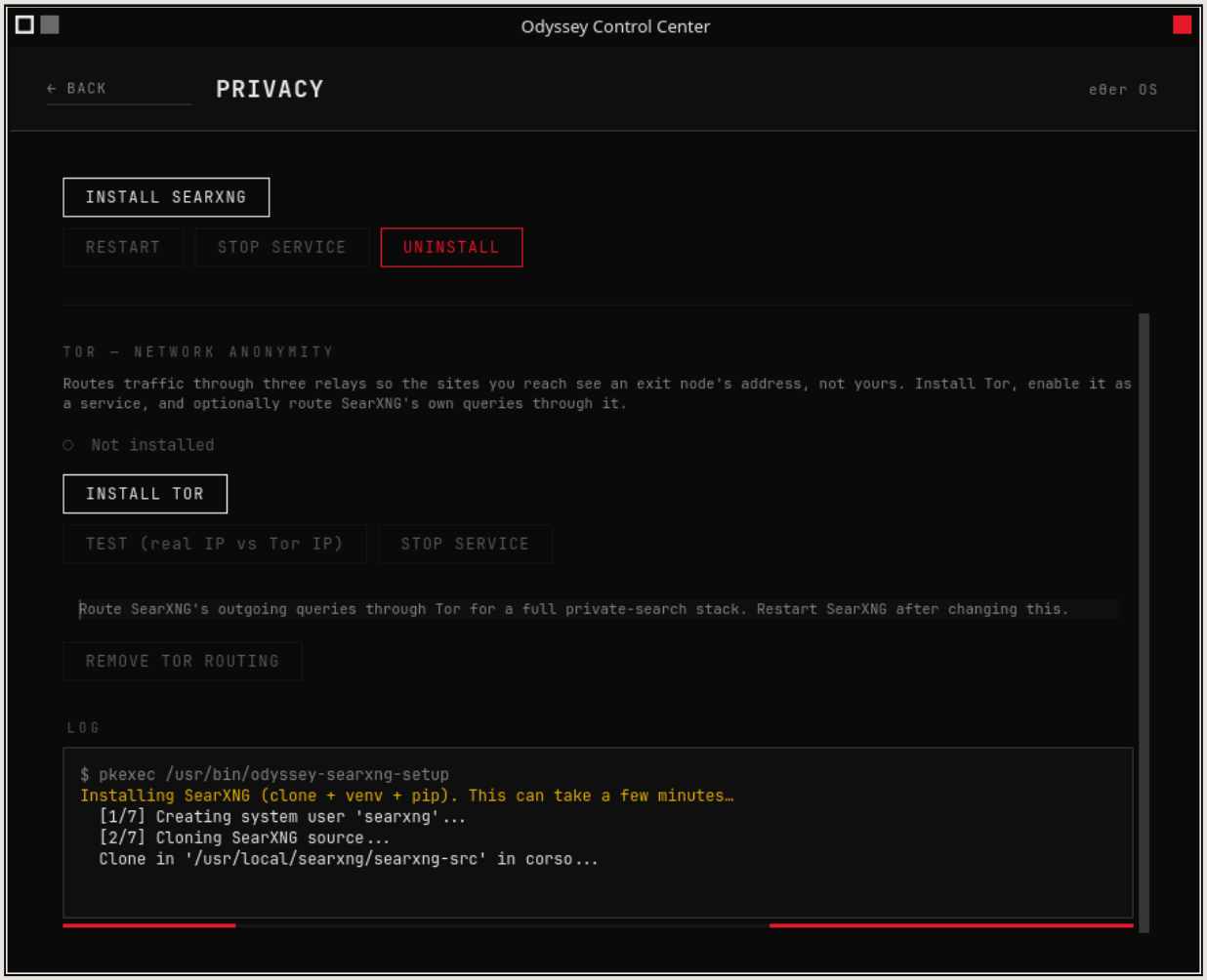

FIG · Mitten in der Installation: das Protokoll überträgt echte Ausgabe — „Installing SearXNG (clone + venv + pip). This can take a few minutes…“ mit sichtbaren Schritten 1/7 und 2/7.

2 ENABLE SERVICE

Sobald installiert, wechselt die Anzeige zu Schritt 2 und die Statuszeile wird gelb: „Installed — service not enabled.“ Nur noch eine Schaltfläche: ENABLE SERVICE.

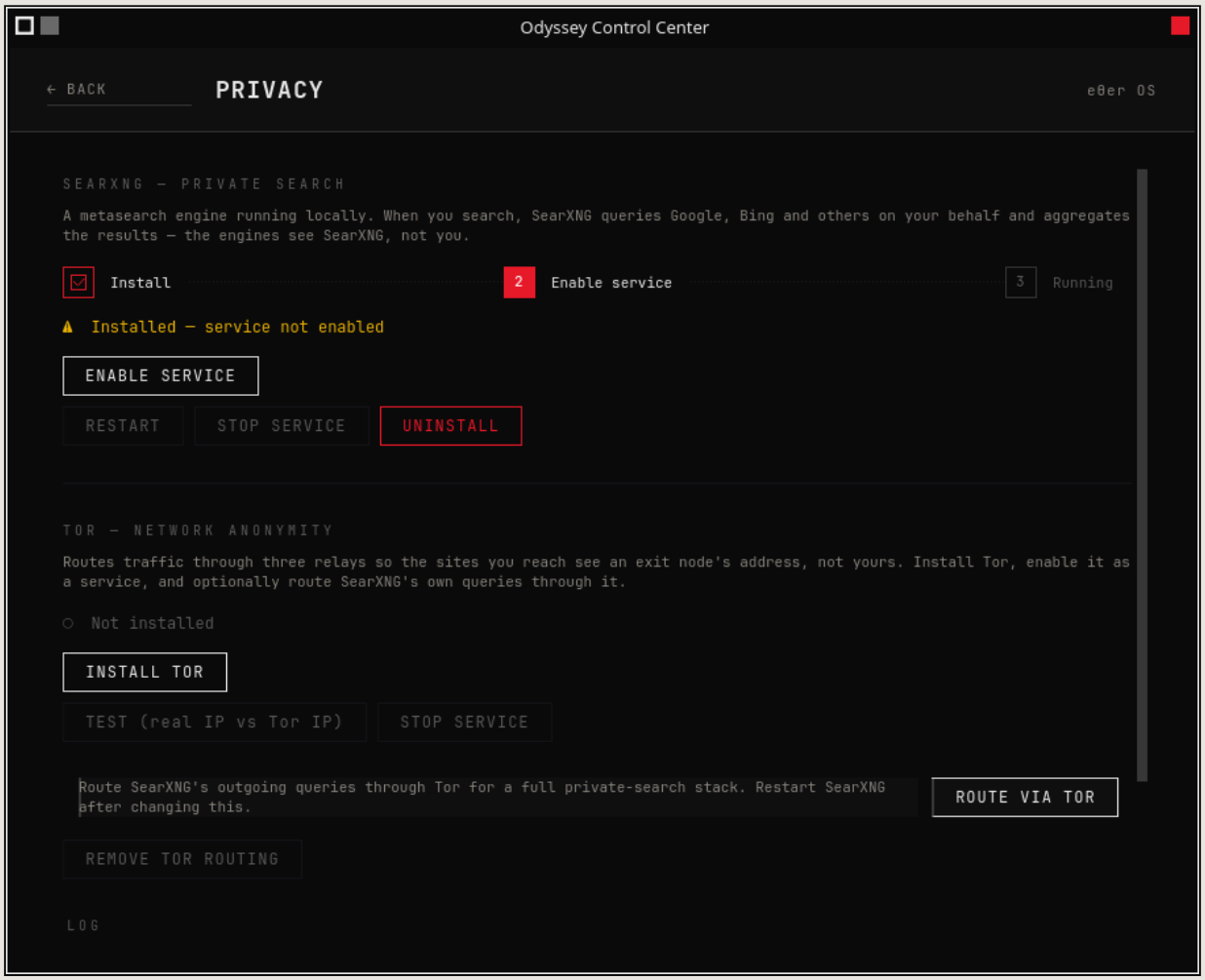

FIG · Schritt 2 aktiv: „Installed — service not enabled“, ENABLE SERVICE als einzige aktive Schaltfläche, RESTART/STOP SERVICE/UNINSTALL noch deaktiviert darunter.

3 Running

Nach dem Aktivieren erreicht die Anzeige Schritt 3 und die gesamte Steuerungszeile wird aktiv. Sobald es läuft, bringt dich **OPEN IN BROWSER** direkt dorthin — SearXNG lauscht standardmäßig auf `http://localhost:8888`.

WHAT	Privacy → INSTALL SEARXNG → ENABLE SERVICE → OPEN IN BROWSER.
WHY	Jede kommerzielle Suchmaschine baut im Laufe der Zeit ein Profil aus deinen Anfragen auf; eine lokale Metasuche-Instanz bedeutet, dass nur du je die nicht aggregierte Anfrage siehst.
RESULT	Eine private Suchmaschine, die auf deiner eigenen Maschine läuft, erreichbar in deinem Browser, die die großen Suchmaschinen in deinem Namen abfragt, ohne dich ihnen direkt auszusetzen.
RISK	Gering — es ist ein lokaler Webdienst wie jeder andere. RESTART, falls er sich seltsam verhält, STOP SERVICE, um ihn zu pausieren, ohne ihn zu entfernen, UNINSTALL, um ihn vollständig zu entfernen.

7.2 Visuelle Anleitung — Tor, Schritt für Schritt

Dieselbe dreistufige Form, unabhängige Anzeige, direkt unter dem SearXNG-Abschnitt.

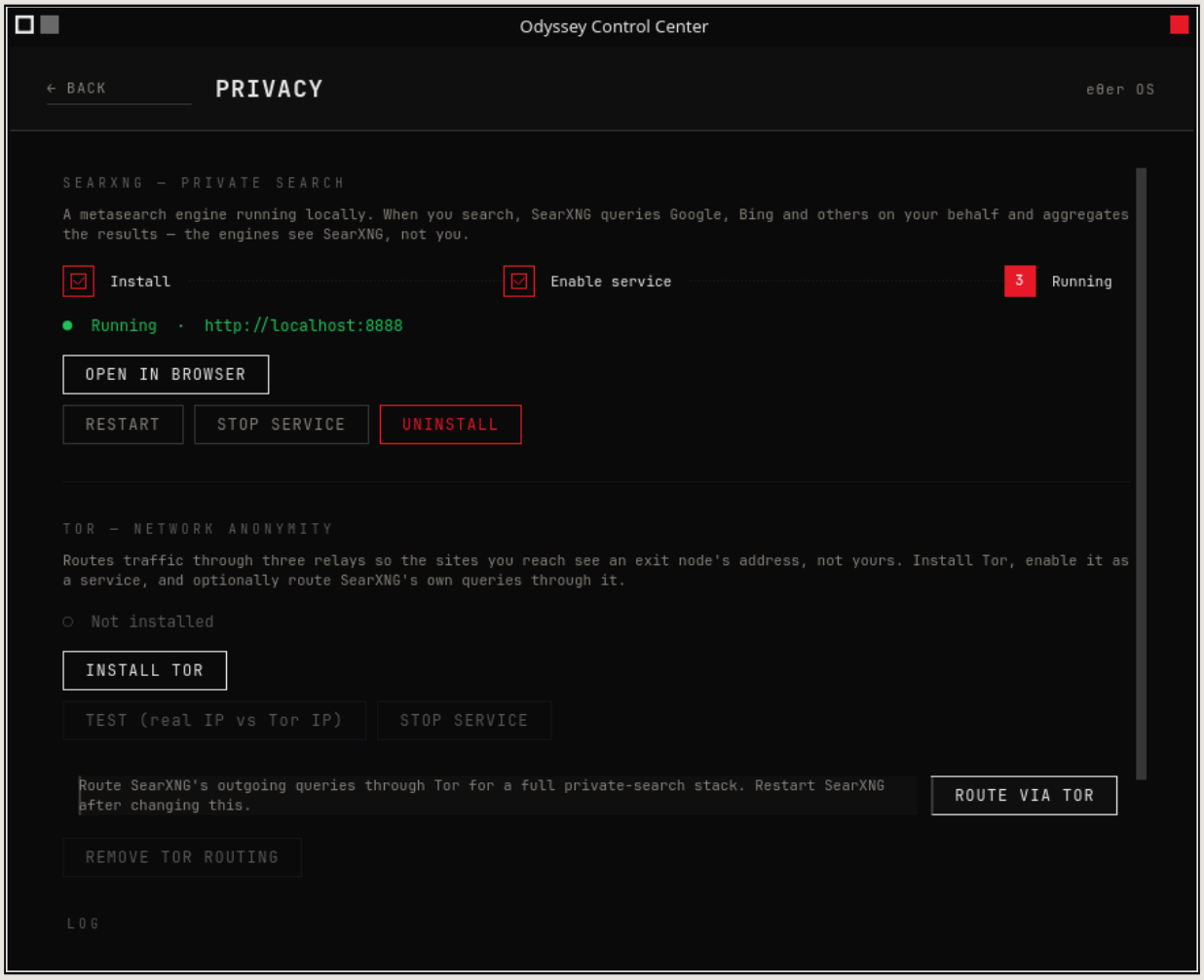

FIG · SearXNG oben vollständig laufend; der Tor-Abschnitt darunter noch bei „Not installed“ mit INSTALL TOR als einziger aktiver Schaltfläche.

Tor leitet deinen Traffic durch drei Relays, bevor er sein Ziel erreicht, sodass die Seite, die du besuchst, die Adresse des Exit-Relays sieht, nie deine eigene.

1 **INSTALL TOR**

Das Protokoll bestätigt die Paketinstallation und zeigt den Dienst in runit verknüpft.

2 **ENABLE SERVICE**

Der Status wechselt zu „Installed — service not enabled“ — dasselbe Muster wie bei SearXNG.

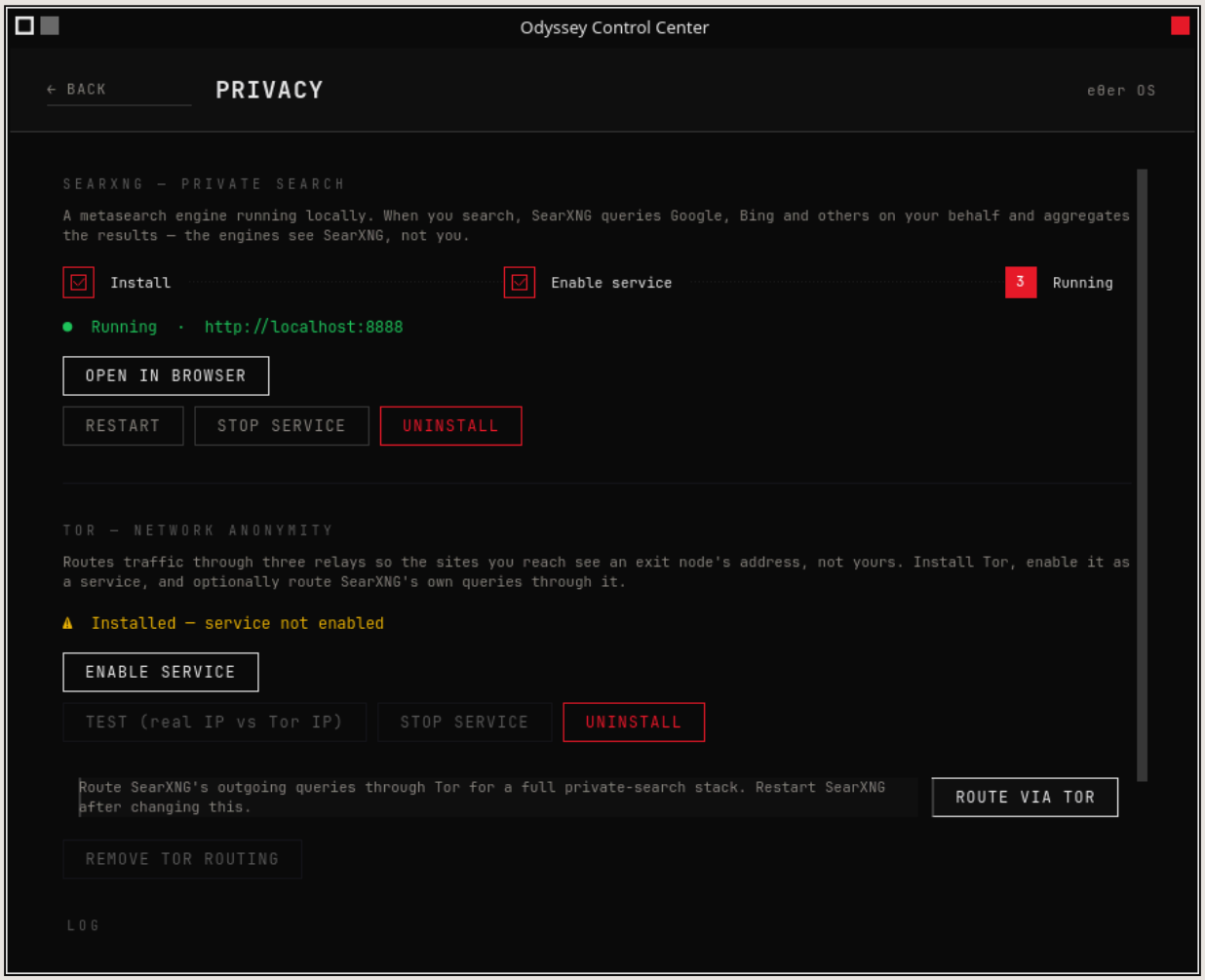

FIG · Tor bei Schritt 2: „Installed — service not enabled“, ENABLE SERVICE aktiv, TEST und STOP SERVICE noch deaktiviert.

3 Running, und überprüfe es

Sobald aktiviert, erscheint eine dedizierte Schaltfläche **TEST (real IP vs Tor IP)** — nutze sie. Sie vergleicht deine echte Adresse mit der, die Tor präsentiert, nebeneinander, sodass du einen tatsächlichen Beweis hast, dass es funktioniert hat, statt einer Statusanzeige, der du nur glauben musst.

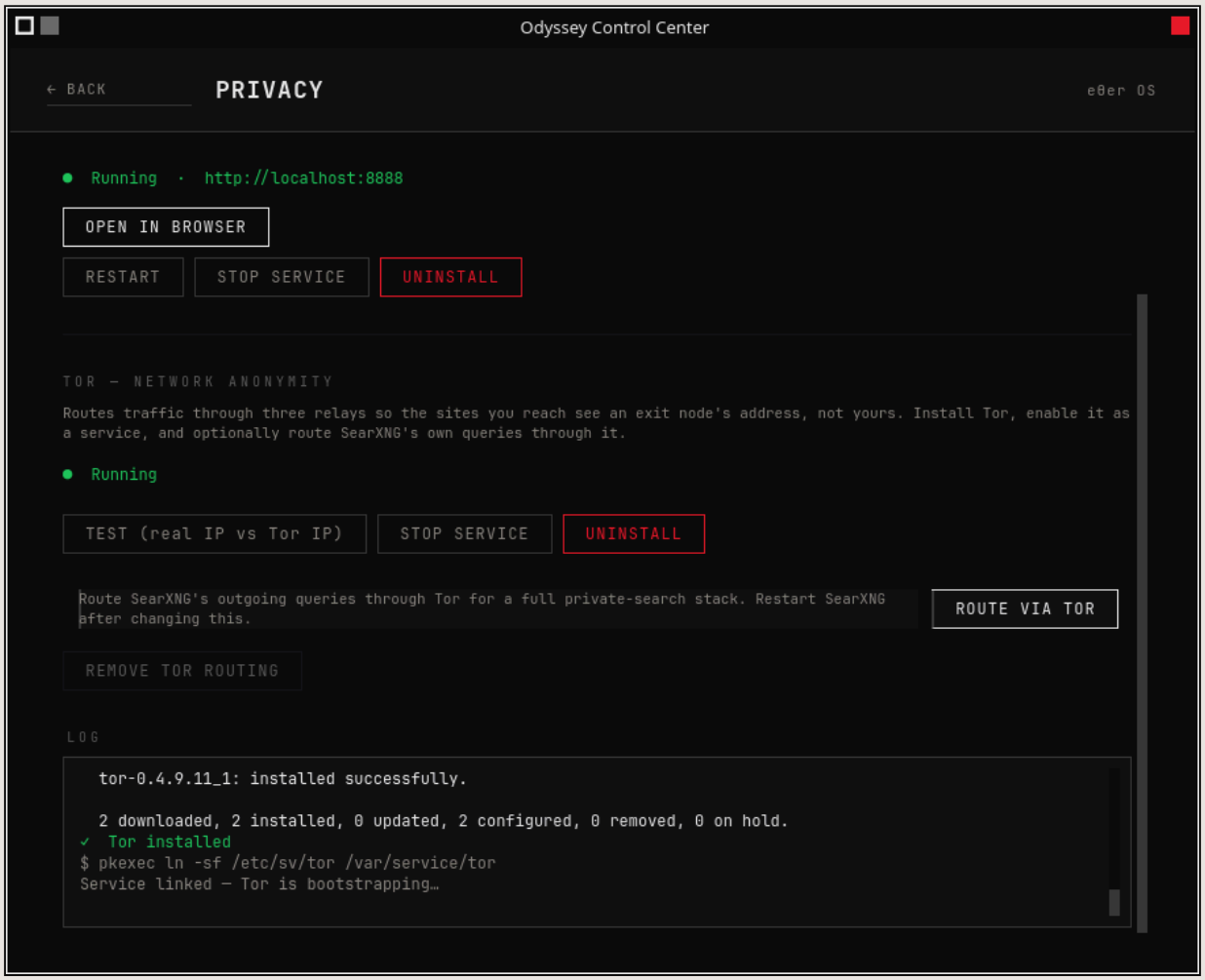

FIG · Beide Abschnitte grün und laufend: SearXNG auf localhost:8888 mit OPEN IN BROWSER, Tor laufend mit TEST / STOP SERVICE / UNINSTALL alle aktiv.

WHAT	Privacy → INSTALL TOR → ENABLE SERVICE → TEST (real IP vs Tor IP) zum Bestätigen.
WHY	Sensibles Surfen über Tor zu leiten verbirgt deine Adresse vor den Seiten, die du besuchst — TEST gibt dir einen Beweis, nicht nur eine Behauptung.
RESULT	Ein laufender Tor-Dienst, über den andere Anwendungen sich verbinden können; TEST zeigt zwei tatsächlich unterschiedliche Adressen, wenn es funktioniert.
RISK	Traffic über Tor ist merklich langsamer — das ist der Preis für die Anonymität. STOP SERVICE pausiert es; UNINSTALL entfernt es.

7.3 Visuelle Anleitung — sie kombinieren

Sobald beide laufen, erscheint ein Hinweis, der anbietet, auch SearXNGs **eigene** Anfragen über Tor zu leiten — nicht nur deine Anfragen an SearXNG, sondern SearXNGs Anfragen an Google und Bing in deinem Namen.

WHAT	Privacy → der Hinweis „Route SearXNG's outgoing queries through Tor" → ROUTE VIA TOR → RESTART im SearXNG-Abschnitt, damit es wirksam wird.
WHY	Ohne das fragt SearXNG selbst weiterhin die großen Suchmaschinen direkt ab — privat von deiner Seite, sichtbar von ihrer. SearXNGs eigenen Traffic über Tor zu leiten schließt diese Lücke für einen vollständig privaten Such-Stack.
RESULT	SearXNGs ausgehende Anfragen, nicht nur deine Anfragen an es, laufen über Tor.
RISK	Ergebnisse laden merklich langsamer, aus demselben Grund wie jeder über Tor geleitete Traffic. REMOVE TOR ROUTING macht es rückgängig, wieder gefolgt von einem Neustart.

▸ **ADVANCED**

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

7.4 Advanced

WAS INSTALL SEARXNG TATSÄCHLICH AUSFÜHRT

```
$ pkexec /usr/bin/odyssey-searxng-setup
# erstellt einen dedizierten Systembenutzer 'searxng',
# klont den Quellcode nach /usr/local/searxng/searxng-src,
# baut ein Python-venv und installiert die Abhängigkeiten mit pip
```

DIENSTE UND PORTS, VON HAND

```
# SearXNGs eigener runit-Dienst
$ sudo sv status searxng
$ sudo sv restart searxng
# Tors runit-Dienst — der Aktivierungsschritt des Panels ist buchstäblich dies
$ sudo ln -sf /etc/sv/tor /var/service/tor
$ sudo sv status tor
# Überprüfen, dass Tor tatsächlich leitet
$ curl -socks5 localhost:9050 https://check.torproject.org/api/ip
```

SEARXNG VON HAND ÜBER TOR LEITEN

Tors SOCKS-Proxy lauscht standardmäßig auf `127.0.0.1:9050`. SearXNG darüber zu leiten bedeutet, seinen ausgehenden HTTP-Client auf diesen Proxy zu richten — die Einstellung lebt in SearXNGs eigener Datei `settings.yml`, unter der Konfiguration ausgehender Anfragen, statt in einer systemweiten Proxy-Variable. Die Schaltfläche ROUTE VIA TOR des Panels bearbeitet diese Datei direkt und startet den Dienst neu; von Hand bedeutet das, denselben `outgoing:`-Block in `/usr/local/searxng/searxng-src/searx/settings.yml` zu finden und zu bearbeiten.

WARUM SEARXNG NACH ROUTING-ÄNDERUNGEN EINEN NEUSTART BRAUCHT

SearXNG liest seine Proxy-Konfiguration nur einmal, beim Start — es ist keine zur Laufzeit neu ladbare Einstellung. Das gilt, egal ob du sie über das Panel oder von Hand änderst, weshalb beide Wege im selben RESTART-Schritt enden.

DATEIEN, DIE DIESES PANEL BERÜHRT

PFAD	ROLLE
`/usr/local/searxng/searxng-src/`	SearXNGs eigene Installation — Quellcode, venv, Konfiguration
`/etc/sv/searxng`, `/etc/sv/tor`	runit-Dienstdefinitionen
`/var/service/`	Wo das Aktivieren eines Dienstes ihn in den laufenden Überwachungsbaum verknüpft

CHAPTER 8

Containers

Container-Verwaltung von Anfang bis Ende: Engine-Einrichtung, eine Liste laufender Container, ein kuratierter Katalog zum Einstieg, rohe Befehle für alles, was nicht darin ist, und Festplattennutzung — alles in einem einzigen scrollbaren Panel.

8.1 Visuelle Anleitung — die Engine

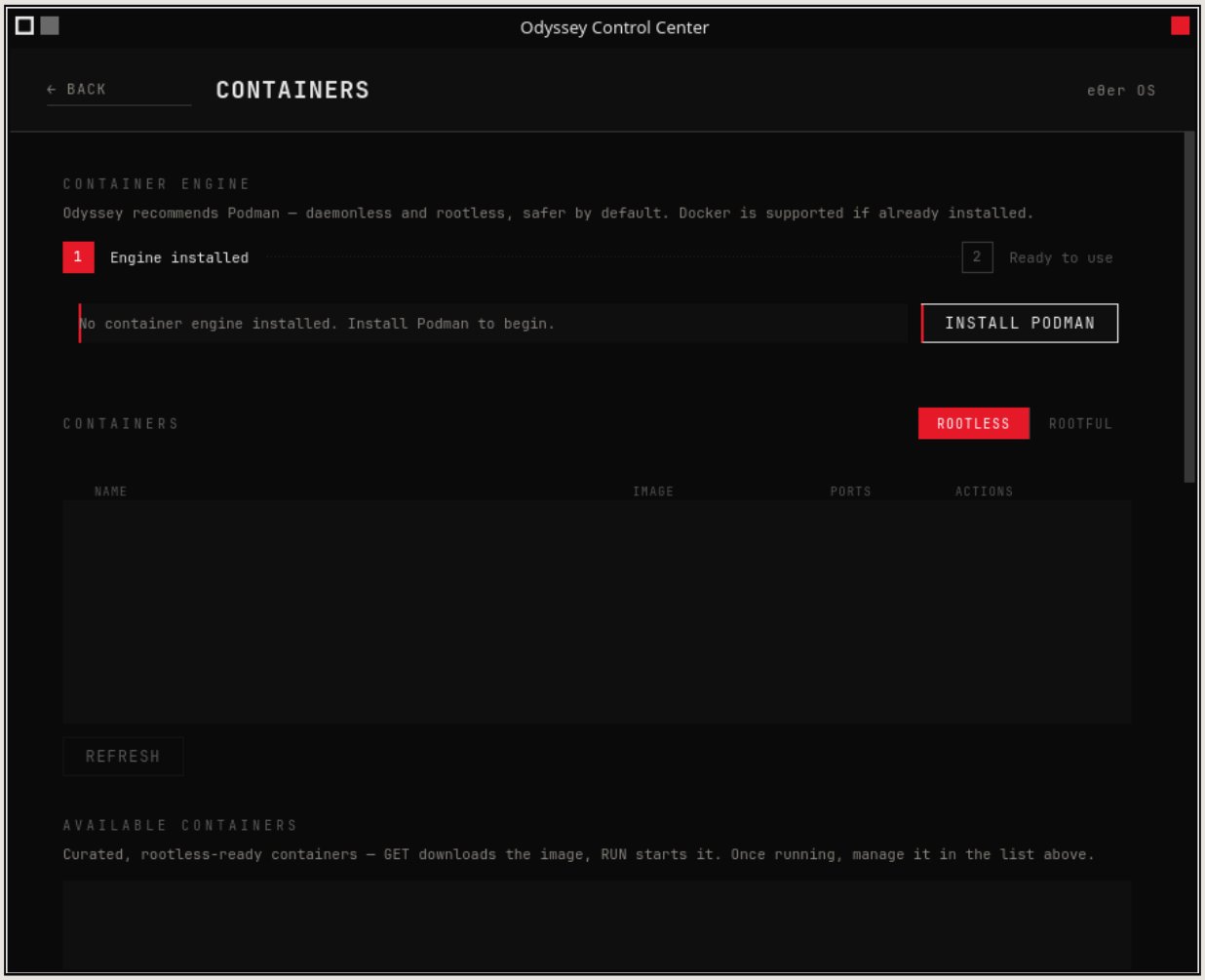

FIG · Schritt 1 von 2: „No container engine installed. Install Podman to begin“, mit INSTALL PODMAN als einziger aktiver Aktion. Die Abschnitte Containers list und Available containers darunter sind sichtbar, aber im leeren Zustand.

Odyssey empfiehlt **Podman** — dämonlos und rootless, was bedeutet, dass Container ohne dauerhaften Hintergrunddienst und standardmäßig ohne Root-Rechte laufen. **Docker** wird ebenfalls unterstützt, falls es bereits das ist, was du benutzt.

1 INSTALL PODMAN

Das Protokoll zeigt die tatsächliche Paketinstallation.

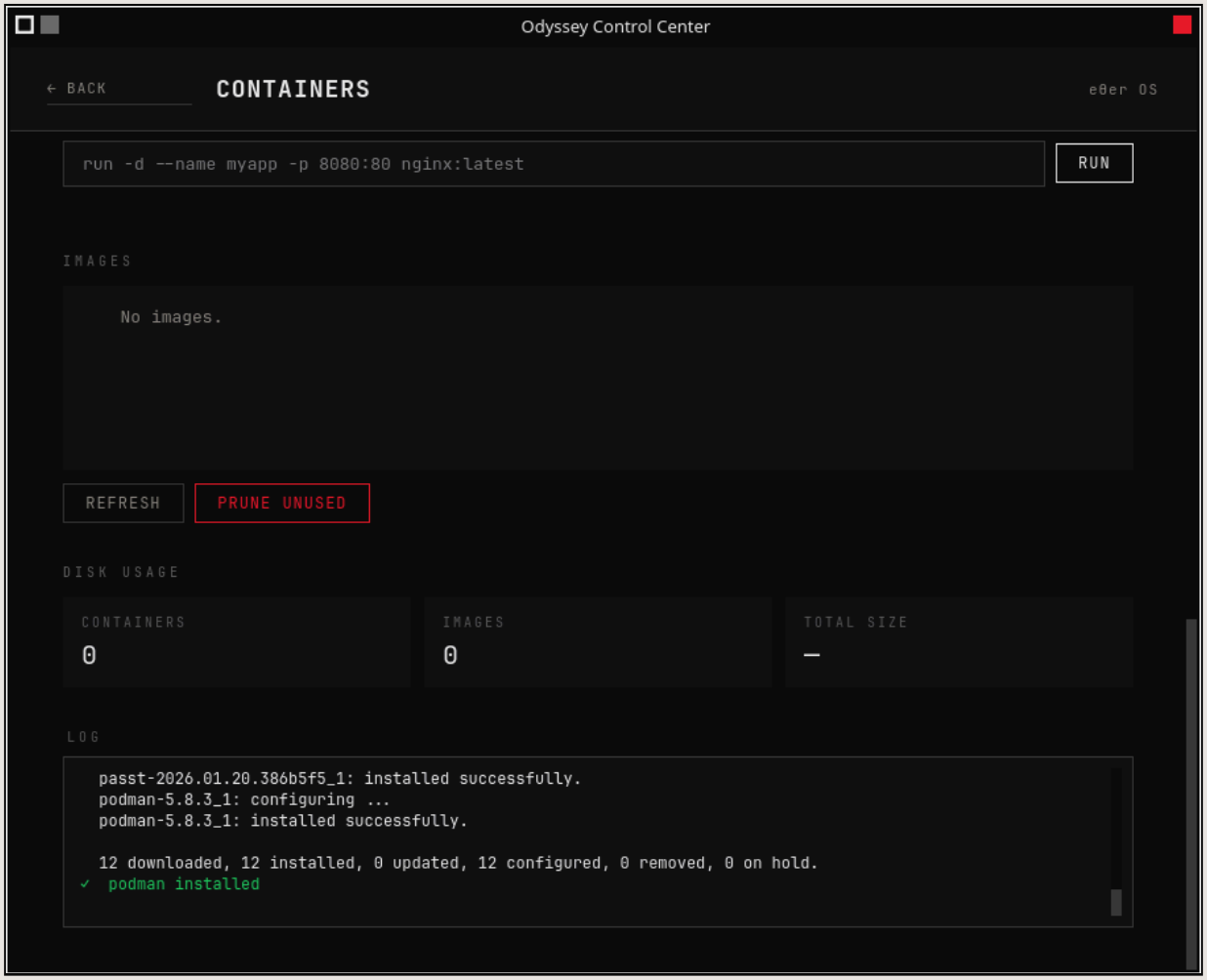

FIG · Nach der Installation: „Engine installed“ rot abgehakt, Schritt 2 „Ready to use”, die Infozeile „podman version 5.8.3 · Podman installed · Docker not installed“, und eine Schaltfläche INSTALL DOCKER TOO für alle, die auch Docker parallel wollen.

2 Ready to use

Die Engine ist aktiv. **INSTALL DOCKER TOO** bleibt verfügbar, wenn du beide Engines nebeneinander willst — sie koexistieren ohne Konflikt.

8.2 Visuelle Anleitung — rootless vs. rootful, und der leere Zustand

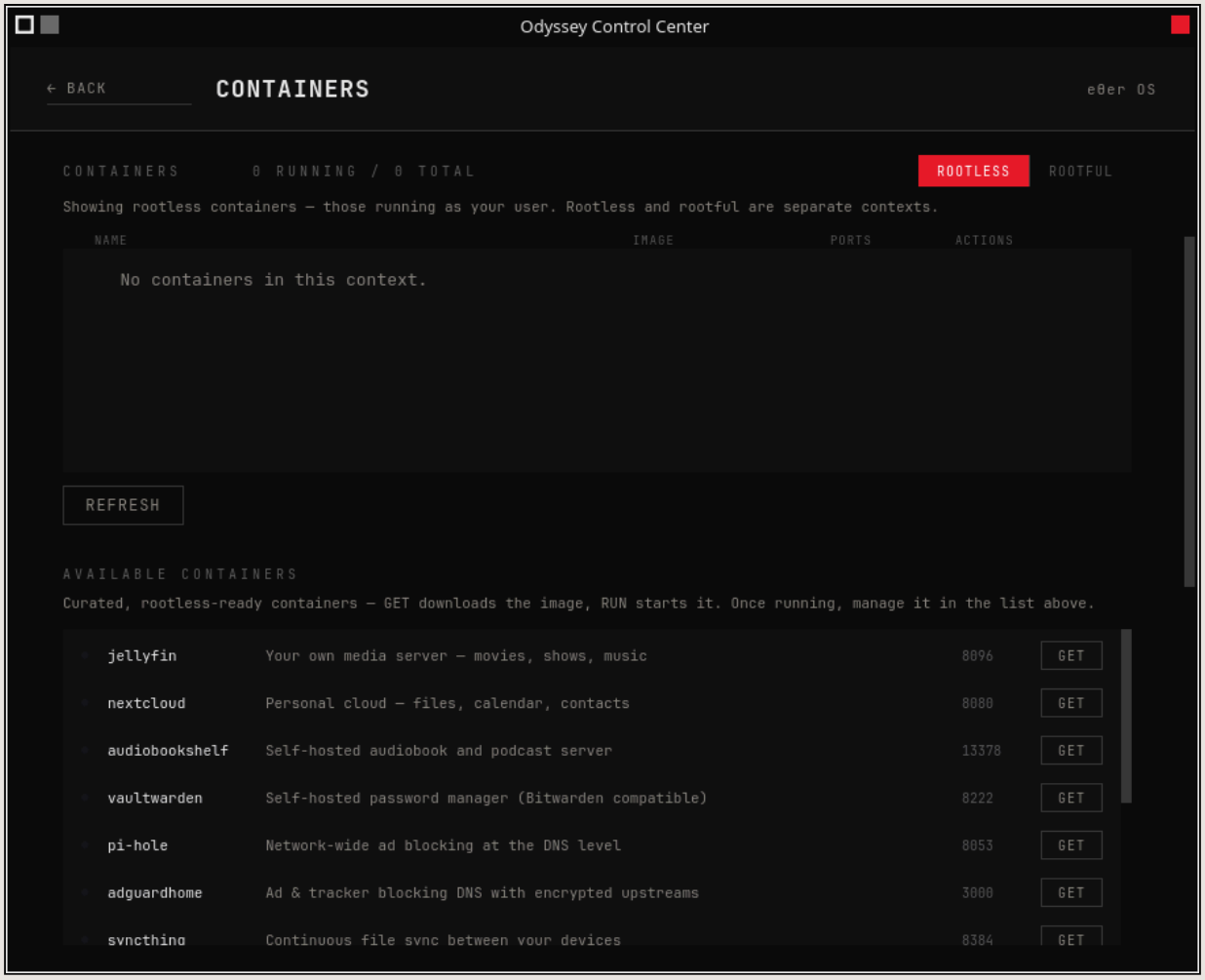

FIG · ROOTLESS ausgewählt (rot hervorgehoben) neben ROOTFUL, „0 RUNNING / 0 TOTAL“, und „No containers in this context.” unter den Spaltenüberschriften NAME / IMAGE / PORTS / ACTIONS.

Der Schalter **ROOTLESS / ROOTFUL** oben wechselt, welchen Kontext du betrachtest und verwaltest — die beiden sind vollständig getrennte Container-Namensräume, kein Filter auf einer kombinierten Liste. Odyssey zeigt standardmäßig ROOTLESS, und diese Voreinstellung ist bewusst gewählt:

WELCHES DER BEIDEN SOLLTEST DU VERWENDEN

Rootless ist die sicherere Wahl und die, zu der dich das Panel lenkt: ein Container, der der Beschränkung im Rootless-Modus entkommt, kann trotzdem nichts tun, was dein eigenes Benutzerkonto nicht bereits könnte. Rootful existiert für die spezifischen Fälle, die es brauchen — bestimmte Netzwerk-Ports mit niedriger Nummer, gewisser Hardwarezugriff — und sollte die Ausnahme sein, nicht die Gewohnheit.

8.3 Visuelle Anleitung — der kuratierte Katalog, von Anfang bis Ende

Scrolle zu **Available containers** — eine bereits fertige Liste, damit das Panel nie eine leere Box ist, die darauf wartet, dass du Image-Namen bereits auswendig kennst. Jede Zeile zeigt den Container, eine Beschreibung in einer Zeile, und den Port, den er benutzt.

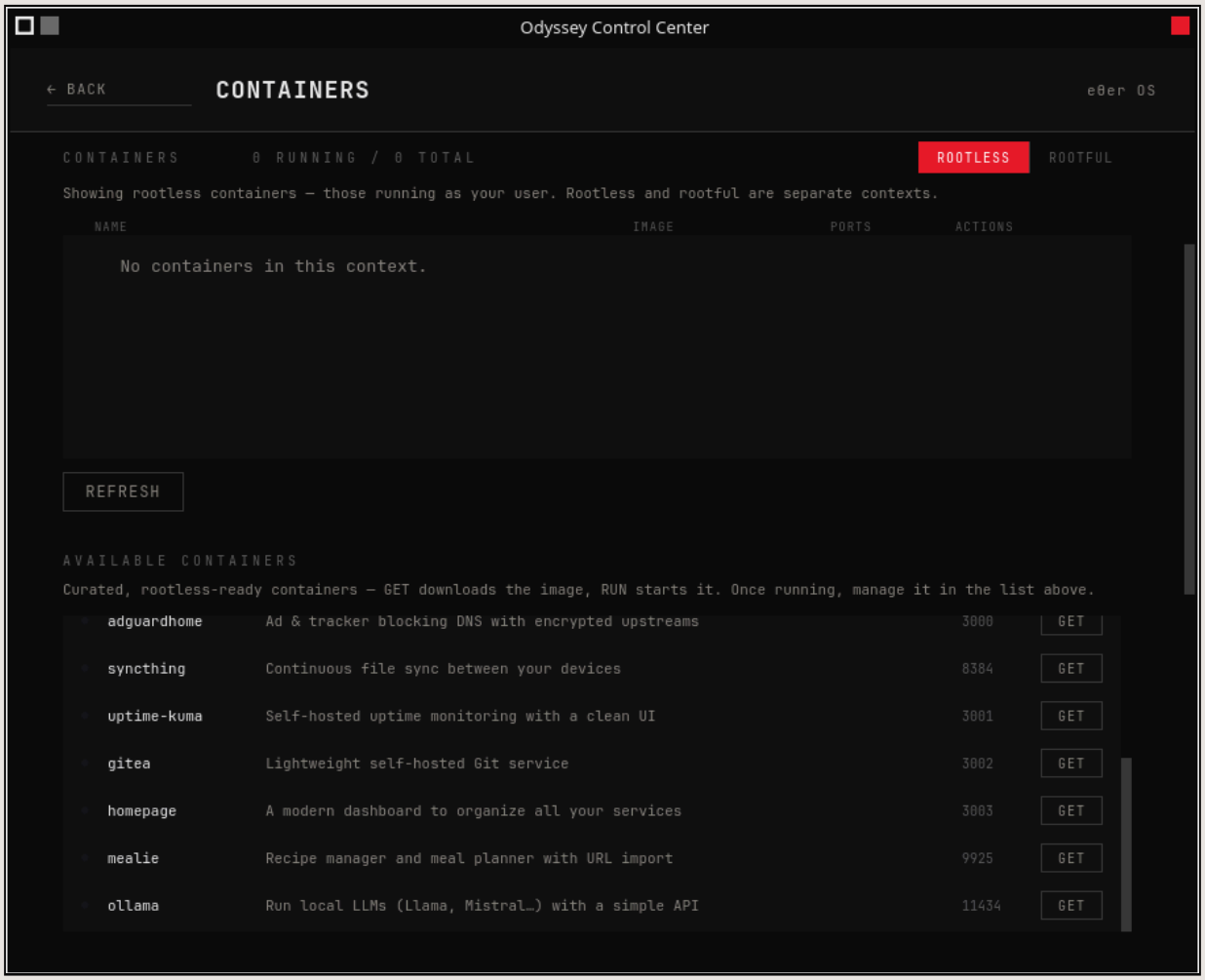

FIG · Die Katalogliste: adguardhome (Port 3000), syncthing (8384), uptime-kuma (3001), gitea (3002), homepage (3003) — mit einer GET-Schaltfläche in jeder Zeile.

Diese Portnummer ist die, unter der du den Dienst erreichst, sobald er läuft — `homepage` mit `3003` bedeutet zum Beispiel `http://localhost:3003`, sobald es aktiv ist.

1 **GET**

Klicke auf GET bei `homepage`. Das lädt nur das Container-Image herunter — noch läuft nichts.

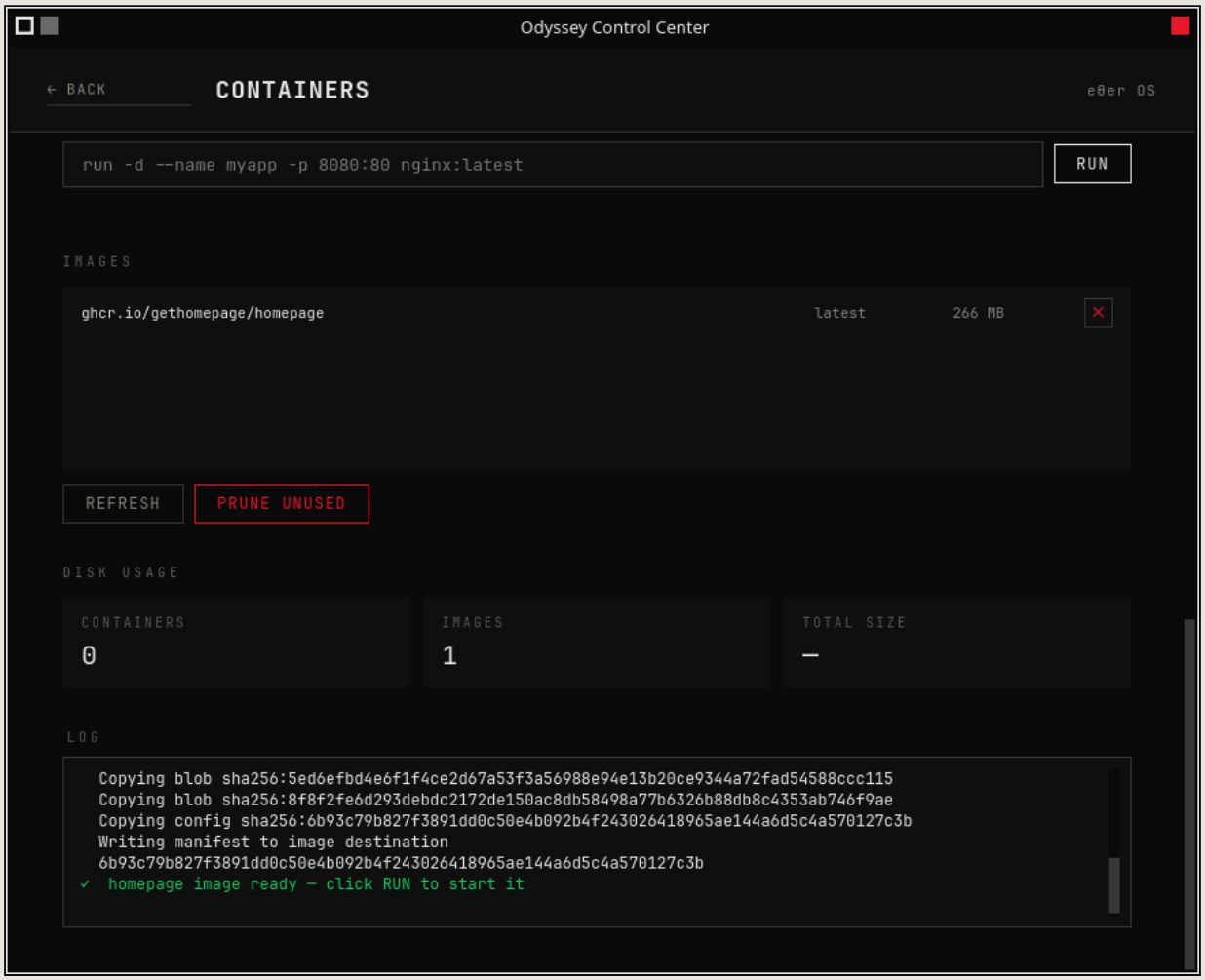

FIG · Das Protokoll überträgt einen echten Image-Download: „Copying blob sha256:5ed6…“, „Writing manifest to image destination”, endend in Grün — „homepage image ready — click RUN to start it.“

2 RUN

Die Zeile der GET-Schaltfläche zeigt jetzt stattdessen RUN — klicke darauf, um den Container wirklich zu starten.

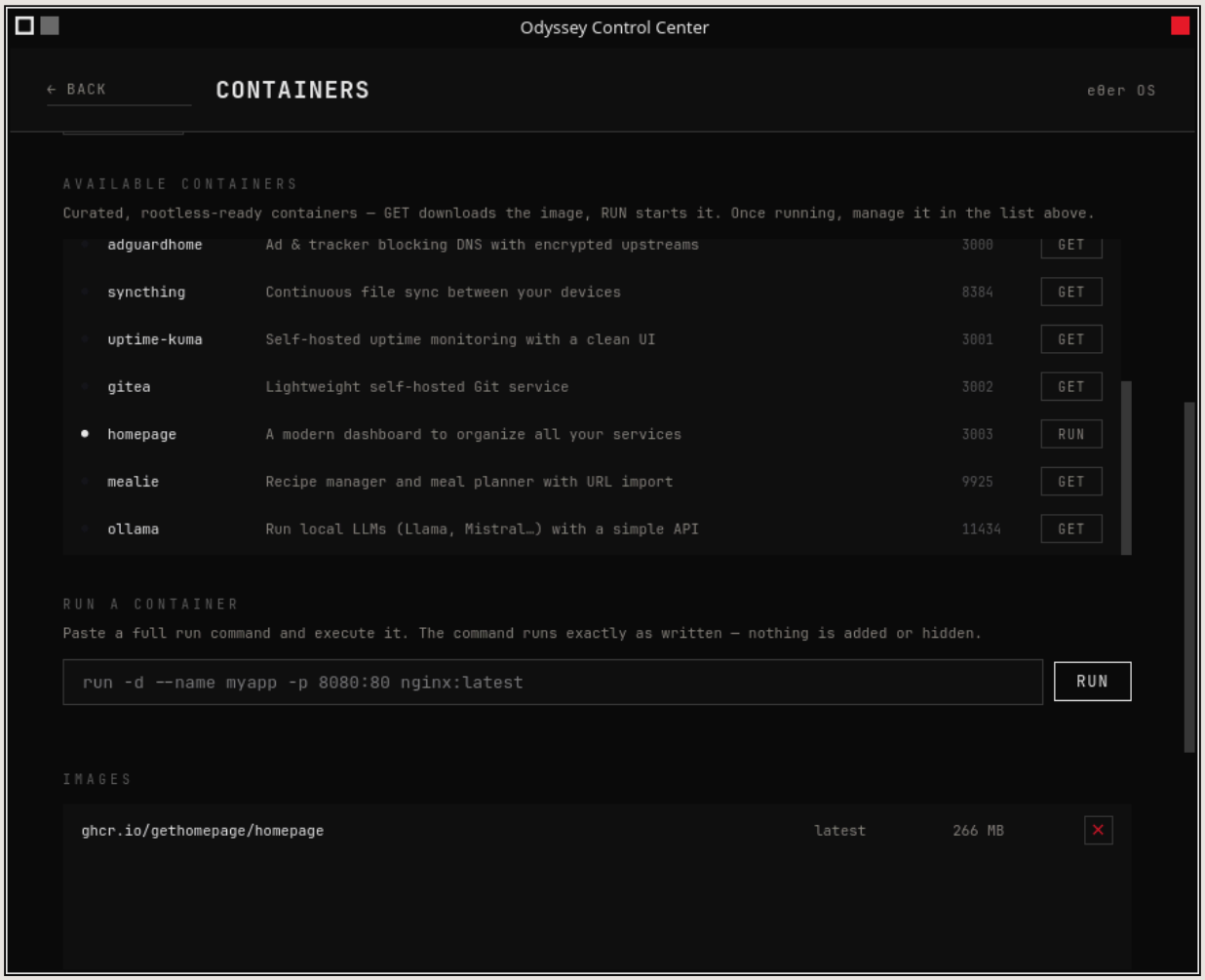

FIG · homepage jetzt oben im Abschnitt CONTAINERS gelistet, grüner Punkt, Image-Pfad ghcr.io/gethomepage/homepage, Ports „0.0.0.0:3003→…“, mit eigener Zeile von Aktionssymbolen; noch sichtbar darunter in Available Containers mit einer neuen RUN-Schaltfläche, um eine weitere Instanz zu starten.

Sobald es läuft, erscheint `homepage` an zwei Stellen gleichzeitig: als aktiver Eintrag in der Liste **Containers** oben (mit seinem Port und Aktionssymbolen), und weiterhin im Katalog darunter, falls du später eine zweite Instanz starten möchtest.

WHAT	Containers → Available containers → GET bei etwas aus dem Katalog → RUN, sobald es heruntergeladen ist.
WHY	Die Aufteilung in zwei Schritte (herunterladen, dann starten) bedeutet, dass du ein Image im Voraus holen kannst, ohne dich sofort zum Ausführen zu verpflichten — nützlich bei langsamer Verbindung oder wenn du nur durchsiehst, was verfügbar ist.
RESULT	Ein laufender Container, sichtbar in der Hauptliste mit seiner Port-Zuordnung, ab dann von dort aus verwaltbar.
RISK	Gering — die selbst gehosteten Dienste des kuratierten Katalogs sind standardmäßig als rootless-freundlich und risikoarm ausgewählt. Die Exposition ist dieselbe wie bei jedem laufenden Webdienst: welcher Port auch immer lauscht, ist gemäß deinen Firewall-Einstellungen erreichbar.

8.4 Visuelle Anleitung — die Container-Liste und ihre Aktionen

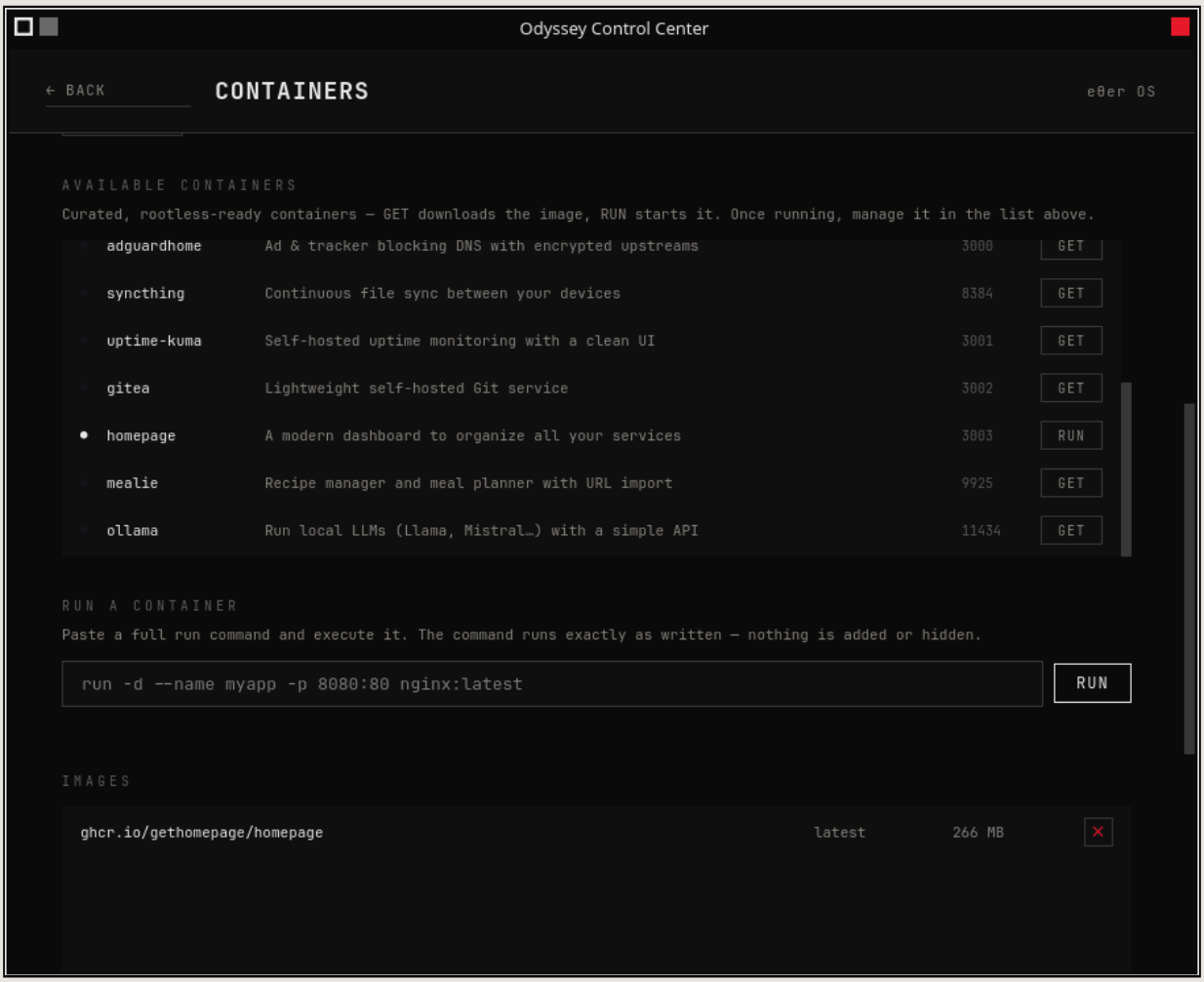

FIG · Die Zeile von homepage in der Containers-Liste, mit ihren fünf Aktionssymbolen rechts sichtbar — die unten erklärten.

Jede Zeile eines laufenden Containers endet in einer Reihe von Symbolen — fahre über jedes mit der Maus, um seine Beschriftung zu sehen, aber hier ist, was jedes tut:

SYMBOL	AKTION
öffnen (pfeil)	Öffnet den Dienst direkt in deinem Browser — der schnelle Weg, sobald etwas läuft
stoppen / starten (quadrat / dreieck)	Stoppt den Container, ohne ihn zu entfernen, oder startet ihn erneut
neu starten (kreispfeil)	Stoppt und startet ihn in einer Aktion — die Lösung für „läuft, antwortet aber nicht“
protokolle (linien-symbol)	Öffnet die eigene Protokollausgabe des Containers — der erste Ort, wo man nachsieht, wenn etwas nicht funktioniert
entfernen (×)	Entfernt den Container vollständig — das Image selbst bleibt heruntergeladen, separat unter Images gelistet

8.5 Visuelle Anleitung — einen Container ausführen, und ihn öffnen

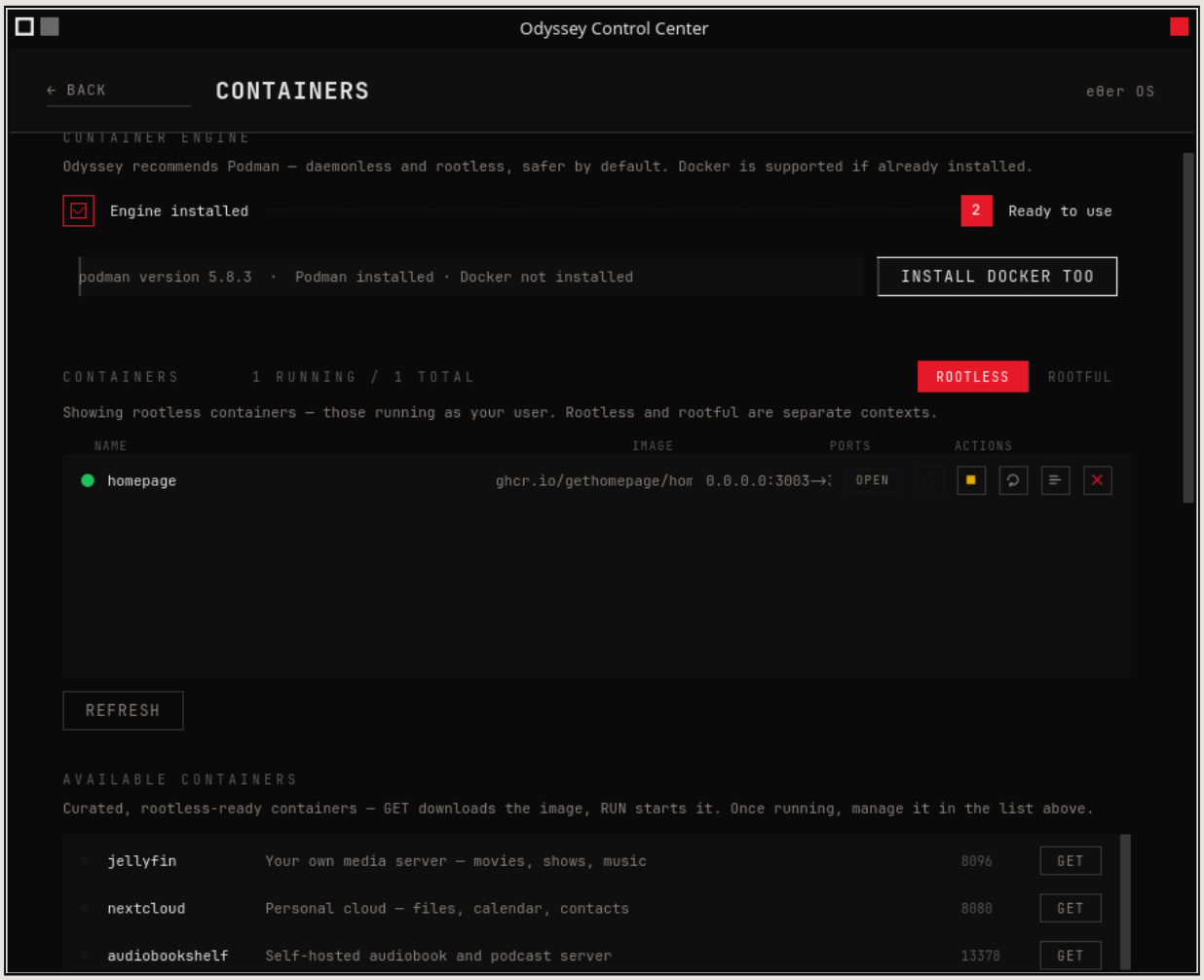

FIG · Das Feld „Run a container“ mit einem Beispielbefehl als Platzhaltertext, die RUN-Schaltfläche, und der Abschnitt Images darunter mit ghcr.io/gethomepage/homepage · latest · 266 MB und einer Entfernen-Schaltfläche.

Für alles, was nicht im Katalog ist, nimmt **Run a container** einen vollständigen `podman run`- (oder `docker run`-)Befehl entgegen und führt ihn genau so aus, wie er geschrieben ist — nichts wird hinzugefügt, nichts versteckt.

WHAT	Containers → „Run a container“ → füge einen vollständigen Run-Befehl ein (z. B. `run -d --name myapp -p 8080:80 nginx:latest`) → RUN.
WHY	Der kuratierte Katalog deckt gängige Fälle ab; das ist der Ausweg für ein bestimmtes Image, bestimmte Flags, bestimmte Ports, die der Katalog nicht bietet.
RESULT	Der Container startet genau so, wie es der Befehl angibt, und tritt der Hauptliste Containers bei, sobald er läuft.
RISK	Was auch immer der Befehl selbst tut — das ist eine treue Weiterleitung, keine Sandbox. Behandle es mit derselben Sorgfalt, mit der du denselben Befehl in einem Terminal eingeben würdest.

Darunter listet **Images** jedes heruntergeladene Image unabhängig davon, ob aktuell ein Container davon läuft, mit **REFRESH** und **PRUNE UNUSED**, um Platz von Images zurückzugewinnen, auf die niemand mehr verweist.

EINEN LAUFENDEN CONTAINER ÖFFNEN

Ein Klick auf OPEN in der Zeile eines laufenden Containers — dieselbe Öffnen-Aktion aus der Symbolzeile oben — startet deinen Standardbrowser direkt bei seiner Adresse und seinem Port, z. B. `http://localhost:3003` für `homepage`. Du musst dir die Portnummer nicht selbst merken, sobald er läuft.

8.6 Festplattennutzung

Ganz unten im Panel, drei Echtzeit-Summen — Container, Images, Gesamtgröße — damit du immer weißt, was dich Container auf der Festplatte kosten, ohne ein Terminal zu öffnen.

```
▸ ADVANCED

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

8.7 Advanced

KOMMANDOZEILEN-ÄQUIVALENTE

```
# was das Panel „GET“ nennt
$ podman pull ghcr.io/gethomepage/homepage
# was das Panel „RUN” bei einem Katalogeintrag nennt
$ podman run -d -name homepage -p 3003:3000 ghcr.io/gethomepage/homepage
# listet laufende Container — rootless-Kontext
$ podman ps
# dasselbe, rootful
$ sudo podman ps
# protokolle, stoppen, neu starten, entfernen — die Symbolzeile, von Hand
$ podman logs -f homepage
$ podman stop homepage
$ podman restart homepage
$ podman rm homepage
# images und festplattennutzung
$ podman images
$ podman system df
$ podman image prune
```

ROOTLESS VS. ROOTFUL, IM HINTERGRUND

Rootless Podman führt Container als dein eigener Benutzer aus, innerhalb eines Benutzer-Namensraums — das „root“ des Containers wird auf einen Bereich privilegienloser UIDs auf dem Host abgebildet, sodass selbst ein vollständig kompromittierter Container-Prozess trotzdem an die Berechtigungen deines eigenen Kontos gebunden bleibt. Rootful Podman (`sudo podman ...`) funktioniert auf traditionelle Weise, mit dem Root des Containers auf das echte Root des Hosts abgebildet. Der Schalter des Panels wechselt eigentlich nur zwischen der Ausführung von einfachem `podman` oder `sudo podman` — wirklich getrennte Zustandsverzeichnisse, getrennte Container-Listen, standardmäßig nichts zwischen den beiden Kontexten geteilt.

```
# wo jeder Kontext seinen eigenen Zustand hält
$ podman info -format '{{.Store.GraphRoot}}'
# rootless:  /.local/share/containers/storage
# rootful: /var/lib/containers/storage
```

WARUM MANCHE KATALOGEINTRÄGE BESTIMMTE OFFENE PORTS BRAUCHEN

Die Port-Zuordnung eines Containers (`3003:3000` im homepage-Beispiel) macht den Dienst standardmäßig nur **von dieser Maschine aus** erreichbar — Podman bindet ihn an `localhost`, sofern nicht anders angegeben. Ihn von einem anderen Gerät in deinem Netzwerk aus zu erreichen, erfordert zusätzlich, dass dieser Port im Firewall-Panel (Kapitel 4) geöffnet ist — die beiden Systeme sind unabhängige Schichten, und ein laufender Container mit zugeordnetem Port ist nicht automatisch aus dem Internet oder dem LAN erreichbar, nur weil Podman darauf lauscht.

CHAPTER 9

Gaming Mode

Ein Hauptschalter oben, eine Echtzeit-Statuszeile, und vier Tabs: **PACKAGES**, **GAMEMODE**, **MANGOHUD**, **PROTON**.

9.1 Visuelle Anleitung — der Hauptschalter

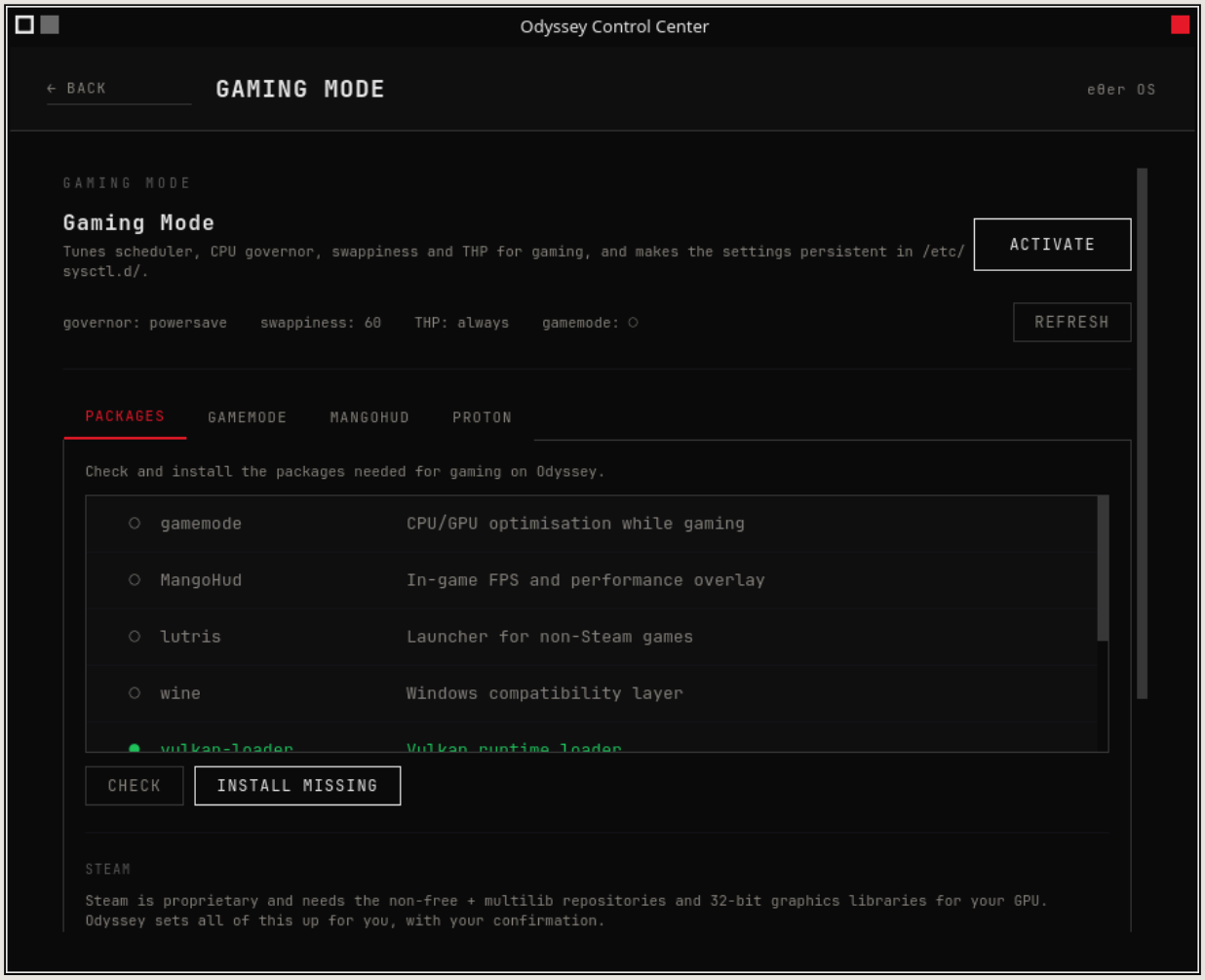

FIG · Der obere Abschnitt des Panels: „Gaming Mode — Tunes scheduler, CPU governor, swappiness and THP for gaming, and makes the settings persistent in /etc/sysctl.d/“, eine Schaltfläche ACTIVATE, eine Schaltfläche REFRESH, und die Live-Statuszeile: governor powersave, swappiness 60, THP always, gamemode off.

Die Statuszeile zeigt dieselben Live-Werte, die auch die Panels Kernel und Memory anzeigen — Governor, Swappiness, THP-Status, und ob der Dienst `gamemode` läuft — sodass du auf einen Blick bestätigen kannst, ob die Aktivierung tatsächlich gewirkt hat, ohne eines der beiden Panels separat zu besuchen.

WHAT	Gaming Mode → ACTIVATE.
WHY	Kernel und Memory manuell aufzusuchen, um von Hand einen Governor, eine Swappiness und einen THP-Wert für Gaming einzustellen, ist genau die Art mehrstufiger Aufgabe, die ein eigener Modus eliminieren sollte.
RESULT	Scheduler, Governor, Swappiness und THP wechseln alle zu gaming-tauglichen Werten, sofort angewendet und dauerhaft gemacht — die Statuszeile aktualisiert sich, um die Änderung widerzuspiegeln.
RISK	Gering, und reversibel — dieselben Werte, die du sonst von Hand in Kernel/Memory einstellen würdest, nur in einem Klick gebündelt.

9.2 Visuelle Anleitung — Packages und Steam

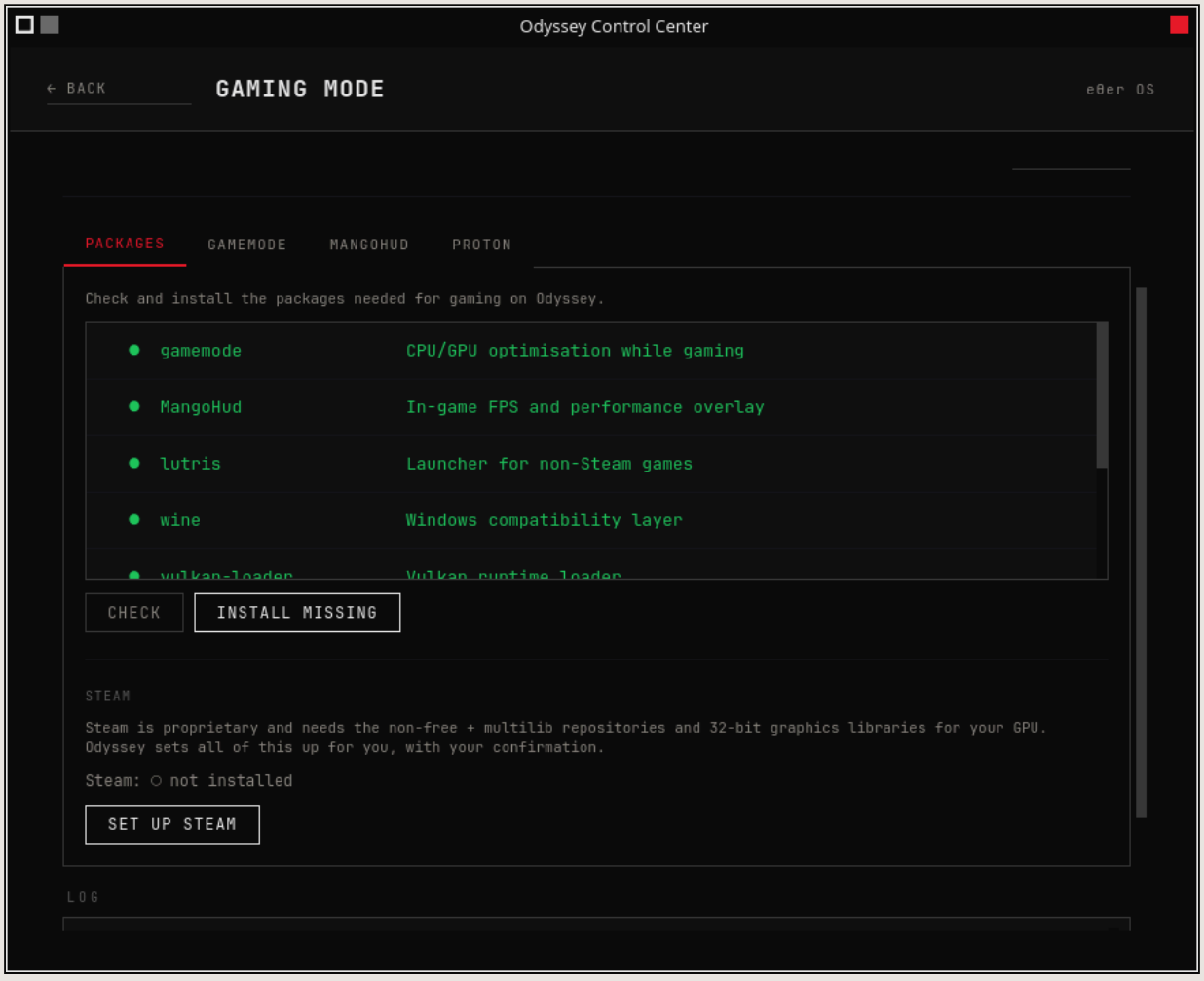

FIG · Der PACKAGES-Tab: gamemode, MangoHud, lutris, wine, vulkan-loader alle mit gefüllten grünen Punkten (installiert); CHECK und INSTALL MISSING oben; der Steam-Abschnitt darunter mit „Steam: ● installed“ und einer REINSTALL-STEAM-Schaltfläche.

Eine Checkliste der für Gaming allgemein benötigten Pakete — CHECK zeigt, was fehlt, INSTALL MISSING schließt die Lücken in einer einzigen Aktion. Sobald alles einen gefüllten grünen Punkt zeigt, wie im Screenshot, bist du vollständig eingerichtet.

Steam hat unterhalb der Checkliste einen eigenen Ablauf, weil es mehr als eine einfache Paketinstallation braucht.

1 **SET UP STEAM**

Klicke darauf — das installiert noch nichts, es öffnet zuerst einen informativen Bestätigungsdialog.

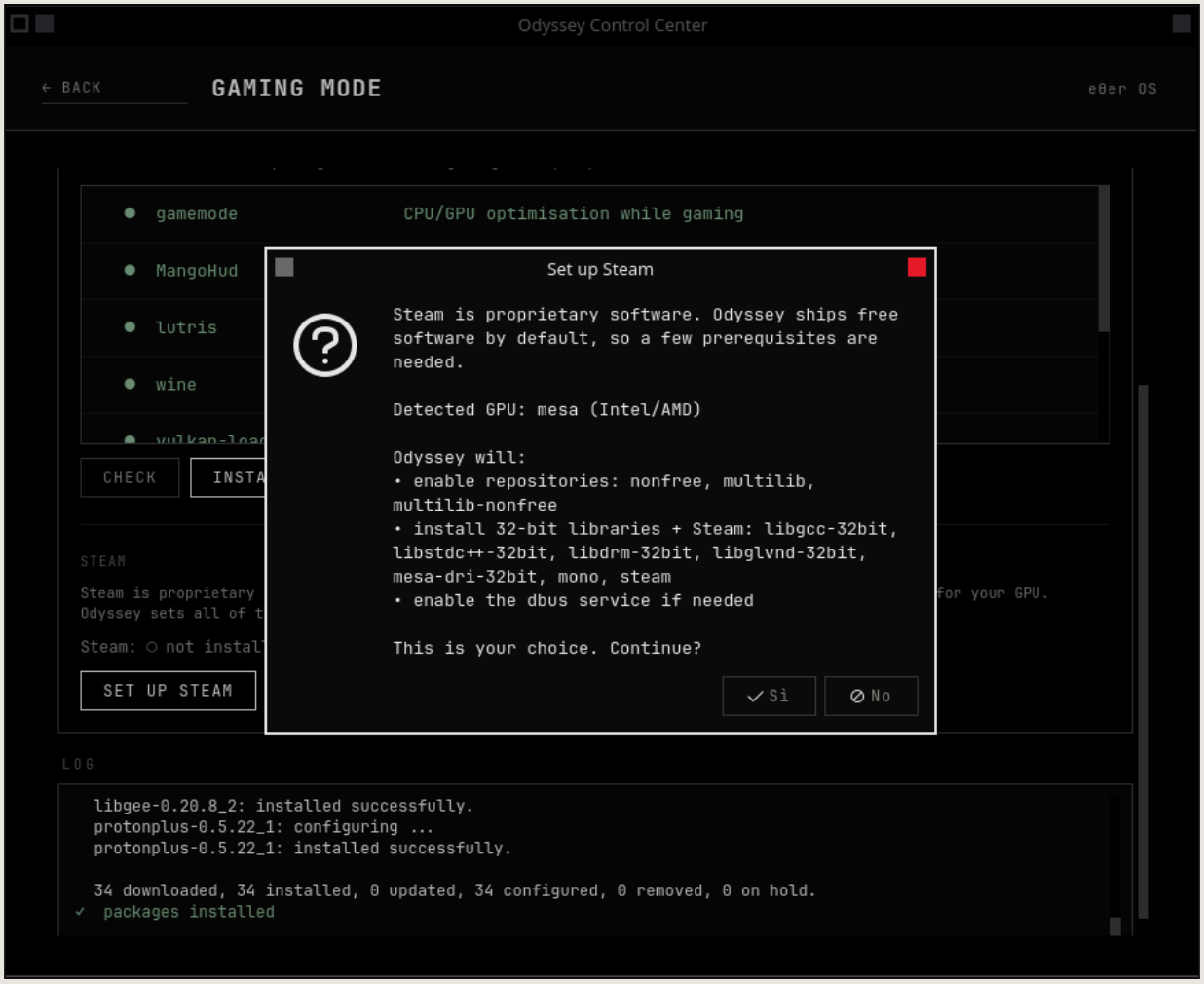

FIG · Der Bestätigungsdialog: „Steam is proprietary software. Odyssey ships free software by default, so a few prerequisites are needed. Detected GPU: mesa (Intel/AMD).“ Eine Aufzählungsliste dessen, was genau aktiviert und installiert wird, endend mit „This is your choice. Continue?” mit den Schaltflächen Sí / No.

Der Dialog sagt dir genau, was passieren wird, bevor du zustimmst: welche Repositories aktiviert werden (`nonfree`, `multilib`, `multilib-nonfree`), welche 32-Bit-Bibliotheken entsprechend deiner **erkannten** GPU installiert werden, und dass der dbus-Dienst aktiviert wird, falls er es nicht bereits ist. Nichts wird hinter der Bestätigung verborgen.

2 **Sí**

Bestätigt es. Das Protokoll überträgt die tatsächliche Installation — Paket-Downloads, Konfiguration, Abschluss.

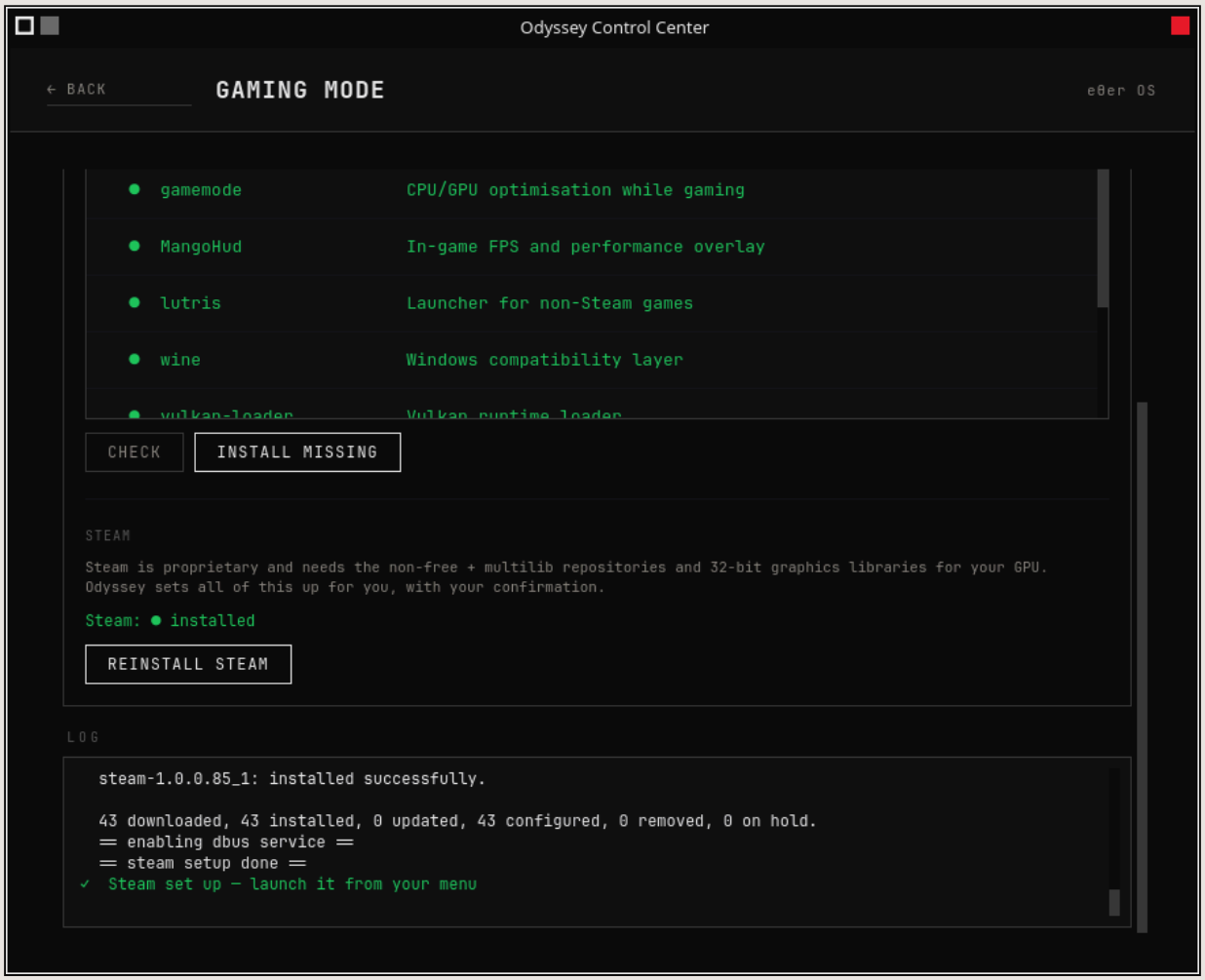

FIG · Der PACKAGES-Tab nach der Einrichtung: alle Pakete grün, Steam zeigt „● installed“ mit REINSTALL STEAM, und das Protokoll endet mit „Steam set up — launch it from your menu.”

WHAT	Gaming Mode → PACKAGES-Tab → SET UP STEAM → prüfe den Dialog → Sí.
WHY	Steam ist proprietär und braucht die non-free/multilib- Repositories sowie GPU-spezifische 32-Bit-Bibliotheken, die Void nicht standardmäßig aktiviert — das von Hand richtig zu machen ist ein häufiges Anfangshindernis für Neulinge auf einem Void-basierten System.
RESULT	Die benötigten Repositories, Steam selbst, und die zu deiner spezifischen GPU passenden 32-Bit-Grafikbibliotheken, alle installiert — starte Steam danach aus deinem Anwendungs-menü.
RISK	Gering — das aktiviert nur Repositories und fügt Pakete hinzu; nichts Bestehendes wird entfernt oder neu konfiguriert. Der Dialog zeigt die vollständige Liste dessen, was passieren wird, bevor du bestätigst.

9.3 GameMode und MangoHud

GameMode (eigener Tab) ist der Pro-Spiel-Leistungsdämon von Feral Interactive — ein durch ihn gestartetes Spiel fordert einen temporären Leistungsschub (CPU-Governor, I/O-Priorität) nur für seinen eigenen Prozess an, automatisch zurückgegeben, wenn das Spiel schließt. **MangoHud** (eigener Tab) ist die Bildschirmüberlagerung, die FPS, Frame-Zeit, CPU-/GPU-Last und Temperaturen während des

Spielens anzeigt. Beide Tabs bieten Installations-/Prüfsteuerelemente und, für MangoHud, schnelle Konfiguration des Erscheinungsbilds der Überlagerung.

9.4 Proton

Verwaltung der Kompatibilitätsschicht zum Ausführen von Windows-Spielen über Steam — Prüfung, was installiert und verfügbar ist, und Installation zusätzlicher Proton-Versionen, einschließlich Community-Builds wie Proton-GE, wenn ein bestimmtes Spiel eine benötigt, die Valve nicht standardmäßig vertreibt.

```
▸ ADVANCED

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

9.5 Advanced

WAS ACTIVATE SCHREIBT

Dieselben zugrundeliegenden Dateien wie die MAKE-PERSISTENT-Aktionen der Panels Kernel und Memory — Gaming Mode ist ein Paket, kein separater Mechanismus:

```
# dauerhaft gemachte sysctl-werte
$ cat /etc/sysctl.d/99-odyssey.conf
# CPU-Governor — kein sysctl, basiert auf sysfs, wird durch
# diesen Mechanismus nicht dauerhaft gemacht; nach dem Start erneut anwenden, falls du
dich darauf verlässt
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

WAS SET UP STEAM TATSÄCHLICH AUSFÜHRT

Der Bestätigungsdialog selbst ist fast eine vollständige technische Offenlegung, aber konkret, auf einem Mesa-System (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
# NVIDIA-Systeme laden zusätzlich nvidia-libs-32bit statt mesa-dri-32bit herunter —
# das Panel erkennt den Hersteller deiner GPU und passt die Paketliste entsprechend an
```

GAMEMODE UND MANGOHUD VON HAND

```
# starte ein beliebiges Spiel mit aktivem GameMode
$ gamemoderun ./game_binary
# oder, für ein Steam-Spiel, Startoptionen:
$ gamemoderun %command%
# MangoHud-Überlagerung, Startoptionen:
$ mangohud %command%
# MangoHuds eigene Konfigurationsdatei, um anzupassen, was die Überlagerung zeigt
$ $EDITOR  /.config/MangoHud/MangoHud.conf
```

EINE BESTIMMTE PROTON-GE-VERSION INSTALLIEREN

Der PROTON-Tab umhüllt `protonplus` oder einen gleichwertigen Versionsmanager — das manuelle Äquivalent ist, eine Version von `GloriousEggroll/proton-ge-custom` auf GitHub herunterzuladen und sie in Steams Verzeichnis für Kompatibilitätswerkzeuge zu entpacken:

```
$ mkdir -p  /.steam/root/compatibilitytools.d
$ tar -xf GE-Proton-VERSION.tar.gz -C  /.steam/root/compatibilitytools.d
# dann pro Spiel in Steam auswählen: Properties → Compatibility
```

CHAPTER 10

Telemetry

Trotz des Namens ist dieses Panel keine Einstellung, die du konfigurierst — es ist ein Echtzeit-Monitor. Odyssey sammelt oder sendet nichts über dich, also gibt es hier nichts abzuschalten. Was dieses Panel stattdessen tut, ist dir in Echtzeit jede ausgehende Verbindung zu zeigen, die dein **eigenes** System herstellt — sodass „mit wem spricht meine Maschine gerade“ etwas ist, das du durch Hinsehen beantworten kannst, nicht indem du einer Behauptung vertraust. Es nennt sich selbst **Reverse Telemetry**, genau deshalb: es dreht die Beobachtung um und richtet sie in deinem Namen auf das System selbst.

10.1 Visuelle Anleitung

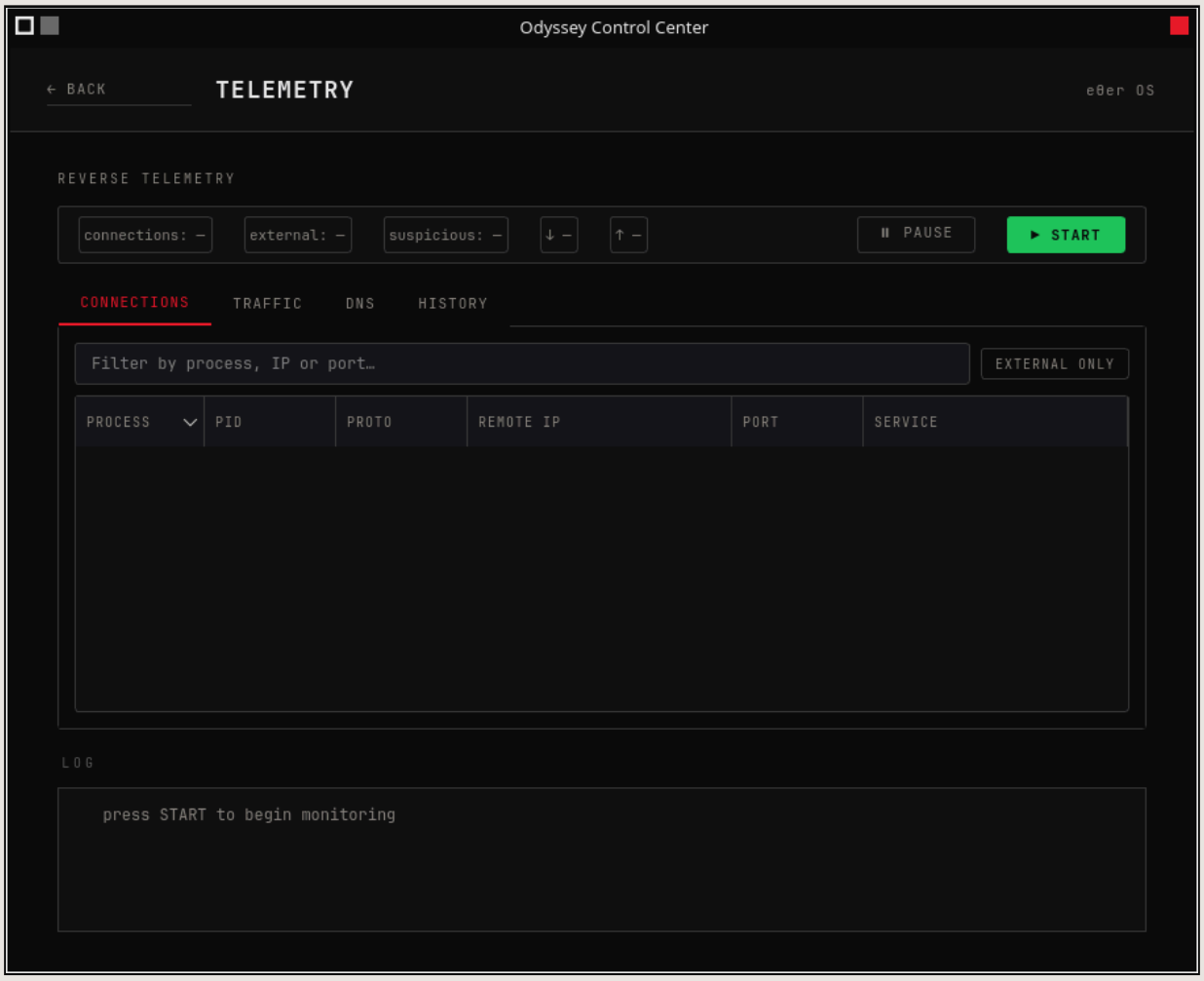

FIG · Vor dem Start: connections/external/suspicious zeigen alle „–“, eine grüne START-Schaltfläche, und das Protokoll sagt „press START to begin monitoring.” Die vier Tabs — CONNECTIONS, TRAFFIC, DNS, HISTORY — bleiben leer über einer unbefüllten Tabelle.

1 START

Startet die Echtzeitüberwachung — die Statusleiste und der aktive Tab füllen sich sofort.

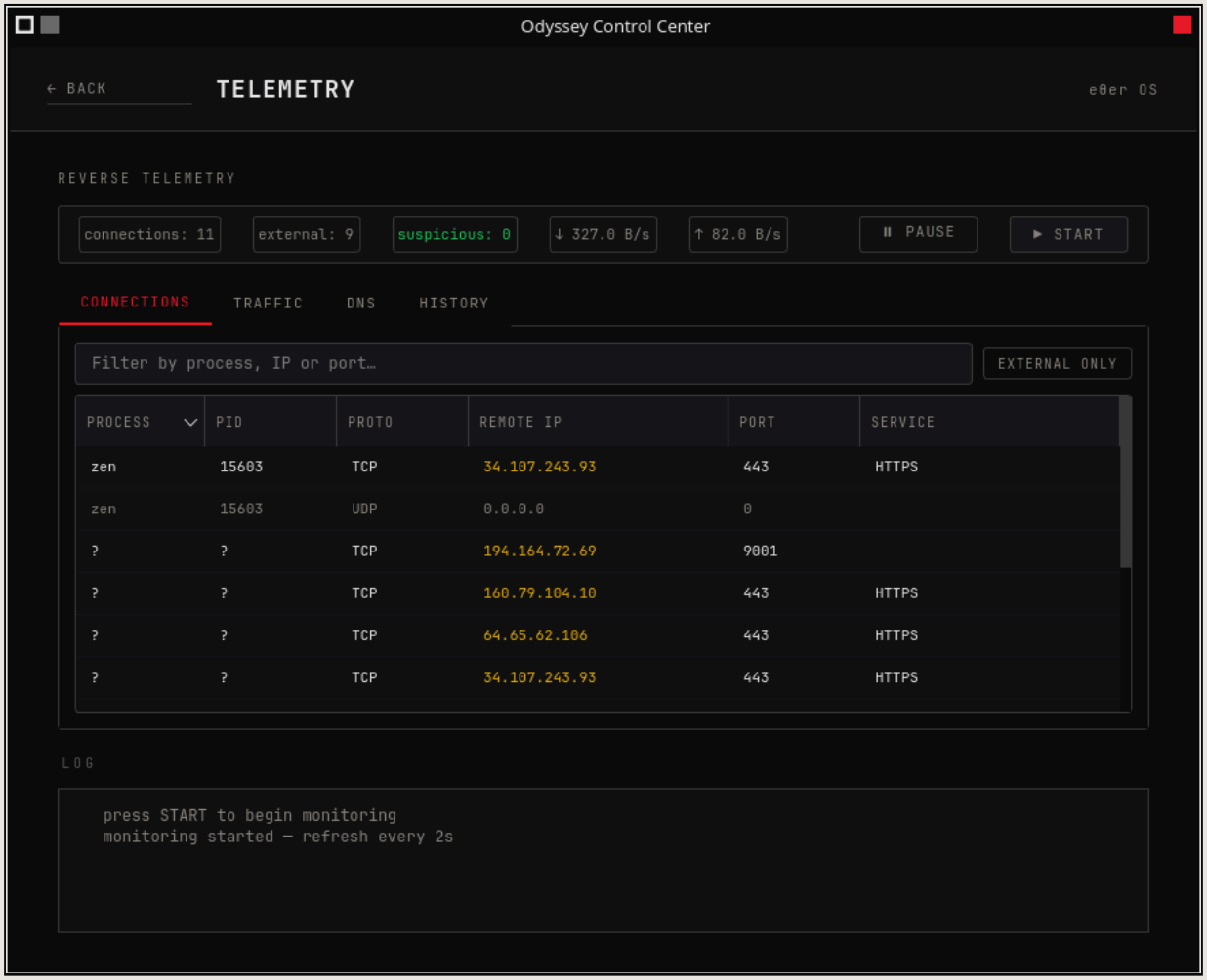

FIG · Nach dem Start: „connections: 11 · external: 9 · suspicious: 0 · ↓327.0 B/s · ↑82.0 B/s“, PAUSE jetzt verfügbar, und die CONNECTIONS-Tabelle voller echter Zeilen: Prozess „zen” über TCP zu 34.107.243.93:443 (HTTPS), mehrere mit „?“ markierte Zeilen für Prozess/PID mit Verbindungen zu anderen Adressen.

Die Statusleiste oben liefert Echtzeitsummen — Verbindungen, extern (vs. lokal), verdächtig, aktuelle Download-/Upload-Geschwindigkeiten — und **PAUSE** friert die Ansicht ein, ohne das Erfasste zu verlieren, **START** setzt fort.

DIE VIER TABS

TAB	WAS ER ZEIGT
CONNECTIONS	Eine Zeile pro aktiver Verbindung: Prozess, PID, Protokoll, entfernte Adresse, Port, und der für diesen Port bekannte Dienst (443 → HTTPS, 22 → SSH). Sortierbar nach jeder Spalte.
TRAFFIC	Durchsatz pro Schnittstelle — aktuelle Download-/Upload-Geschwindigkeit, plus kumulierte Summen seit Beginn der Überwachung.
DNS	Ein Echtzeitstrom der Domainnamensauflösungen, während sie geschehen — oft klarer als eine bloße IP, da ein Hostname dir sagt **was**, nicht nur **wo**.
HISTORY	Alles, was die Sitzung erfasst hat, als scrollbares Protokoll — nützlich, um etwas Kurzes zu erwischen, das keine aktive Verbindung mehr ist, wenn du hinschaust.

Zeilen, in denen der Prozess `?` anzeigt (sichtbar im Screenshot), bedeuten, dass die Verbindung erfasst wurde, aber der Ursprungsprozess in diesem Moment nicht anhand der PID aufgelöst werden konnte — häufig bei sehr kurzlebigen Verbindungen. Ein **Filter**-Feld über der Tabelle grenzt nach Prozess, IP oder Port ein, und **EXTERNAL ONLY** verbirgt rein lokalen Traffic.

WHAT	Telemetry → START → beobachte CONNECTIONS, oder sortiere nach PROCESS oder REMOTE IP, um etwas Bestimmtes zu untersuchen.
WHY	Es ist die direkte, sachliche Antwort auf „erreicht etwas auf dieser Maschine das Netzwerk, das es nicht sollte“ — direkt aus den eigenen Verbindungstabellen des Kernels gelesen, nicht aus der Behauptung eines Anbieters über seine eigene Software.
RESULT	Ein live und präzises Bild jeder ausgehenden Verbindung, kontinuierlich aktualisiert, solange sie läuft und nicht pausiert ist.
RISK	Keins — vollständig nur lesend.

Verbindungen von Prozessen ohne gewöhnlichen Grund, das Netzwerk aus eigenem Antrieb zu erreichen — eine nackte Shell, ein eigenständig laufender Skript-Interpreter — werden automatisch unter „suspicious“ in der Statusleiste markiert, sodass eine Anomalie auffällt, statt im normalen Traffic deines Browsers und der Systemdienste begraben zu werden.

▸ ADVANCED

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

10.2 Advanced

WOHER DIE DATEN KOMMEN

Das Panel liest dieselben Verbindungstabellen auf Kernel-Ebene, die jedes Standardwerkzeug lesen würde, aktualisiert in kurzen Intervallen während die Überwachung aktiv ist — kein eigener spezieller Treiber oder Kernel-Modul. Die Kommandozeilen-Äquivalente:

```
# jeder offene Socket, mit dem besitzenden Prozess — dieselben Felder wie CONNECTIONS
$ sudo ss -tupn
# durchsatz pro schnittstelle — dieselben zahlen wie TRAFFIC
$ ip -s link
# eine einmalige Live-Ansicht, falls du sie außerhalb des Panels willst
$ sudo nethogs
```

DNS-AUFLÖSUNGEN, VON HAND

```
# Live-Erfassung ausgehender DNS-Anfragen (benötigt ein Paketerfassungswerkzeug)
$ sudo tcpdump -i any -n port 53
```

WARUM MANCHE ZEILEN PROCESS ALS „?“ ZEIGEN

Eine Live-Verbindung ihrem besitzenden Prozess zuzuordnen erfordert das Lesen von `/proc/PID/fd` für jeden Kandidatenprozess im Moment der Erfassung — wenn die Verbindung schneller geöffnet und geschlossen wird als das Aktualisierungsintervall des Panels, oder der besitzende Prozess bereits

beendet ist, hat das Panel kein Zeitfenster für diese Zuordnung und zeigt ? statt zu raten. Das ist eine Timing-Beschränkung, kein Zeichen, dass etwas verborgen wird.

CHAPTER 11

About

Ein einziges, umfassendes Diagnoseblatt der Maschine, aktualisiert bei jedem Öffnen des Tabs, damit es immer widerspiegelt, was sich in anderen Panels geändert hat, ohne das Control Center neu öffnen zu müssen.

11.1 Visuelle Anleitung

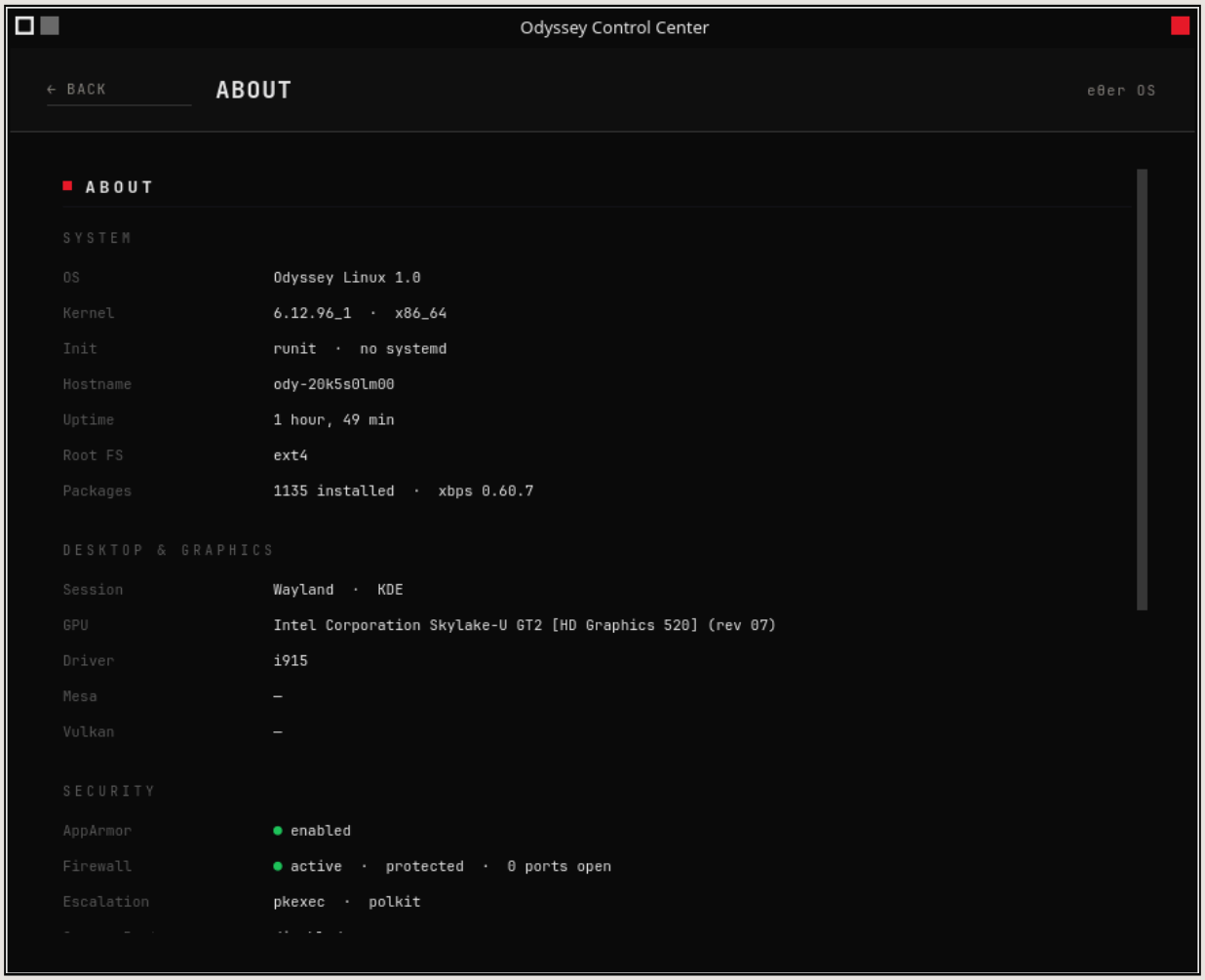

FIG · Das Panel bis nach oben gescrollt: Abschnitt SYSTEM zeigt OS „Odyssey Linux 1.0", Kernel 6.12.96_1 · x86_64, Init runit · no systemd, Hostname, Uptime, Root FS ext4, Packages „1135 installed · xbps 0.60.7"; DESKTOP & GRAPHICS darunter mit Session, GPU, Driver, Mesa, Vulkan; SECURITY beginnt, AppArmor „● enabled" und Firewall „● active · protected · 0 ports open" zu zeigen.

Drei Abschnitte, in dieser Reihenfolge: **SYSTEM** (Kernel, Init, Hostname, Uptime, Dateisystem, Paketanzahl), **DESKTOP & GRAPHICS** (Sitzungstyp, GPU, Treiber, Mesa-/Vulkan-Unterstützung), und weiter

unten **SECURITY** (AppArmor- und Firewall-Status mit farbigen Indikatoren, nicht nur Text — grün für aktiviert/aktiv) und **HARDWARE** (Mainboard, Audio).

WHAT	About → scrolle durch die drei Abschnitte, um jeden Wert zu prüfen — Kernel-Version, verwendeter Treiber, ob AppArmor und die Firewall wirklich aktiv sind.
WHY	Es ist der schnellste Weg, den Status des gesamten Systems an einem Ort zu bestätigen, ohne fünf verschiedene Panels zu öffnen.
RESULT	Ein vollständiger, aktueller Schnappschuss — bei jedem Besuch des Tabs aufgefrischt, sodass er nie veraltete Informationen aus einer früheren Sitzung zeigt.
RISK	Keins — vollständig nur lesend.

11.2 Copy report

Eine einzige Schaltfläche oben im Panel, **COPY REPORT**, legt das gesamte Blatt als Klartext in die Zwischenablage — jeder Abschnitt, jeder Wert — bereit, direkt in einen Forenbeitrag oder einen Eintrag im Bug-Tracker eingefügt zu werden.

WHAT	About → COPY REPORT → in deinen Fehlerbericht oder Forenbeitrag einfügen.
WHY	Fehlerberichte sind nur so nützlich wie die beigefügten Systemdetails, und Kernel-Version, Treiber, AppArmor-Status und ein Dutzend andere Felder von Hand einzutippen ist genau die Art von Reibung, die viele Leute den Schritt überspringen lässt.
RESULT	Ein vollständiger, korrekt formatierter Systembericht in der Zwischenablage, bereit zum Einfügen — dieselben Informationen, die ein Maintainer sonst Feld für Feld erfragen müsste.
RISK	Keins für dein System. Der Bericht ist Diagnoseinformation (Versionen, Status, Hardware-modell) — er enthält keine persönlichen Dateien oder Zugangsdaten — aber wie bei jedem Diagnose-Dump, lies ihn durch, bevor du ihn irgendwo öffentlich postest, wenn du dir nicht sicher bist, was er enthält.

▸ ADVANCED

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

11.3 Advanced

WOHER JEDES FELD KOMMT

Nichts in diesem Panel ist erfunden oder seit dem Installationszeitpunkt zwischengespeichert — jeder Wert wird live abgefragt, jedes Mal, wenn der Tab geöffnet wird:

```
$ 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
```

Abschnitt Security im Speziellen:

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

WARUM ES SICH LOHNT, ZUERST ABOUT ZU PRÜFEN

Da es das System bei jedem Besuch erneut abfragt, ist About der schnellste Weg zu bestätigen, dass eine in einem anderen Panel vorgenommene Änderung wirklich Wirkung gezeigt hat — wende einen Firewall-Modus in Firewall an, prüfe dann den Abschnitt Security von About statt Firewall selbst erneut zu öffnen. Es ist als der Gegencheck „hat das wirklich funktioniert?“ für den Rest des Control Center gebaut.

CHAPTER 12

NVIDIA driver

AUSSTEHENDE INTEGRATION

Dieses Kapitel ist ein Platzhalter. Das NVIDIA-Modul erscheint im Control Center nur auf Maschinen mit erkannter NVIDIA-Hardware, und die Screenshots dafür können nur auf einer solchen Maschine aufgenommen werden — sie werden separat bereitgestellt und dieses Kapitel in einer zukünftigen Überarbeitung dieses Handbuchs vervollständigt. Es steht absichtlich zuletzt, nach About, genau damit das spätere Hinzufügen die Seitennummerierung oder Abschnittsverweise keines anderen Kapitels verschiebt.

Was bereits bekannt ist und das fertige Kapitel verankern wird: das Panel trägt ein sichtbares **BETA**-Abzeichen und einen einleitenden Hinweis, dass der Treiber selbst proprietär ist — von Void aus dem offiziellen Installer von NVIDIA mit fest hinterlegter Prüfsumme gepackt, aber nicht unabhängig überprüfbar, dieselbe Ehrlichkeit, die der Rest von Odyssey auf sich selbst anwendet. Es erkennt deine spezifische GPU durch Abgleich mit einer eingebauten Datenbank und wählt automatisch den richtigen Treiberzweig (current/Turing-und-neuer, oder einen von drei Legacy-Zweigen für ältere Karten, ausweichend auf den Open-Source-Treiber `nouveau`, wenn nichts Proprietäres zutrifft). Die Installation läuft als verfolgte Pipeline in sieben Schritten mit sichtbarer Fortschrittsanzeige, und das Panel druckt seinen eigenen Wiederherstellungsbefehl im Voraus aus — `odyssey-nvidia --revert` von einer Textkonsole aus — noch bevor du auf Installieren klickst. Auf Hybrid-Laptops erscheint automatisch ein Abschnitt PRIME render-offload, der es einem bestimmten Spiel oder Programm erlaubt, auf der NVIDIA-GPU zu laufen, während alles andere für die Akkulaufzeit auf der integrierten GPU bleibt.