ODYSSEY LINUX · DOCUMENTAZIONE DI RIFERIMENTO

Odyssey Control Center — Guida Pratica

Ogni pannello, sezione per sezione. Cosa fa, cosa cambia, e cosa cliccare — scritta per chi non l'ha mai usato prima.

Launch Edition

Don't trust me. Verify me.

— nobody

Odyssey Linux — Odyssey Control Center: Guida Pratica
Launch Edition. Copre il Control Center così come viene distribuito sulla ISO della Launch Edition — dieci pannelli completi, da Boot ad About, più un capitolo segnaposto per il modulo driver NVIDIA (visibile solo su hardware NVIDIA, in attesa degli screenshot da una macchina che lo possiede).

Copyright © 2026 the Odyssey Linux project.
Rilasciata sotto licenza Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).

Questa guida è il complemento pratico della Security and Hardening Guide, che copre i pannelli Firewall e AppArmor in piena profondità tecnica — qui trovi la guida visiva passo passo e un rimando per il resto.

Ogni schermata, pulsante e comportamento descritto qui è stato letto dal codice sorgente reale del Control Center e da screenshot veri, non a memoria.

Etichette dei pulsanti, nomi dei pannelli, percorsi dei file e comandi restano in inglese perché sono l’interfaccia reale del programma: cercarli tradotti nel Control Center non li troveresti.

INDICE

1 Cos’è il Control Center 1
1.1 Pattern che ritrovi in ogni pannello 2
2 Boot 3
2.1 Guida visiva — trovare un altro sistema operativo e impostare il default . . 3
2.2 Guida visiva — scegliere cosa si avvia per default 5
2.3 Guida visiva — la tab EFI 7
2.4 Parametri del kernel 9
2.5 Advanced 9
3 Kernel 11
3.1 Guida visiva — i preset 11
3.2 Guida visiva — SYSCTL, gruppo per gruppo 12
3.3 Guida visiva — CPU governor 14
3.4 Guida visiva — hugepages 15
3.5 Guida visiva — modules 16
3.6 Advanced 17
4 Memory & Swap 19
4.1 Guida visiva 19
5 3,88 GiB su un sistema da 15,52 GiB). 21
5.1 Advanced 21
6 Firewall 23
6.1 Guida visiva 23
6.2 Advanced 24
7 AppArmor 25
7.1 Guida visiva — STATUS 25
7.2 Guida visiva — PROFILES 26
7.3 Guida visiva — COMMUNITY 27
7.4 Guida visiva — VIOLATIONS 28
7.5 Guida visiva — EDITOR 29
7.6 Advanced 30
8 Privacy 32
8.1 Guida visiva — SearXNG, passo per passo 32
8.2 Guida visiva — Tor, passo per passo 34
8.3 Guida visiva — combinarli 37
8.4 Advanced 38
9 Containers 40
9.1 Guida visiva — il motore 40
9.2 Guida visiva — rootless vs. rootful, e lo stato vuoto 42
9.3 Guida visiva — il catalogo curato, dall’inizio alla fine 42
9.4 Guida visiva — la lista dei container e le sue azioni 46
9.5 Guida visiva — eseguire un container, e aprirlo 47
9.6 Uso del disco 48
9.7 Advanced 48
10 Gaming Mode 50
10.1 Guida visiva — l’interruttore principale 50
10.2 Guida visiva — Packages e Steam 51
10.3 GameMode e MangoHud 53
10.4 Proton 54
10.5 Advanced 54
11 Telemetry 56
11.1 Guida visiva 56
11.2 Advanced 58

12 About 60
12.1 Guida visiva 60
12.2 Copy report 61
12.3 Advanced 61
13 NVIDIA driver 63

CHAPTER 1

Cos’è il Control Center

L’Odyssey Control Center sono dieci piccoli programmi indipendenti dietro un unico launcher — uno per area di sistema: Boot, Kernel, Memory & Swap, Firewall, AppArmor, Privacy, Containers, Gaming Mode, Telemetry, About, e (su hardware NVIDIA) il modulo driver NVIDIA.

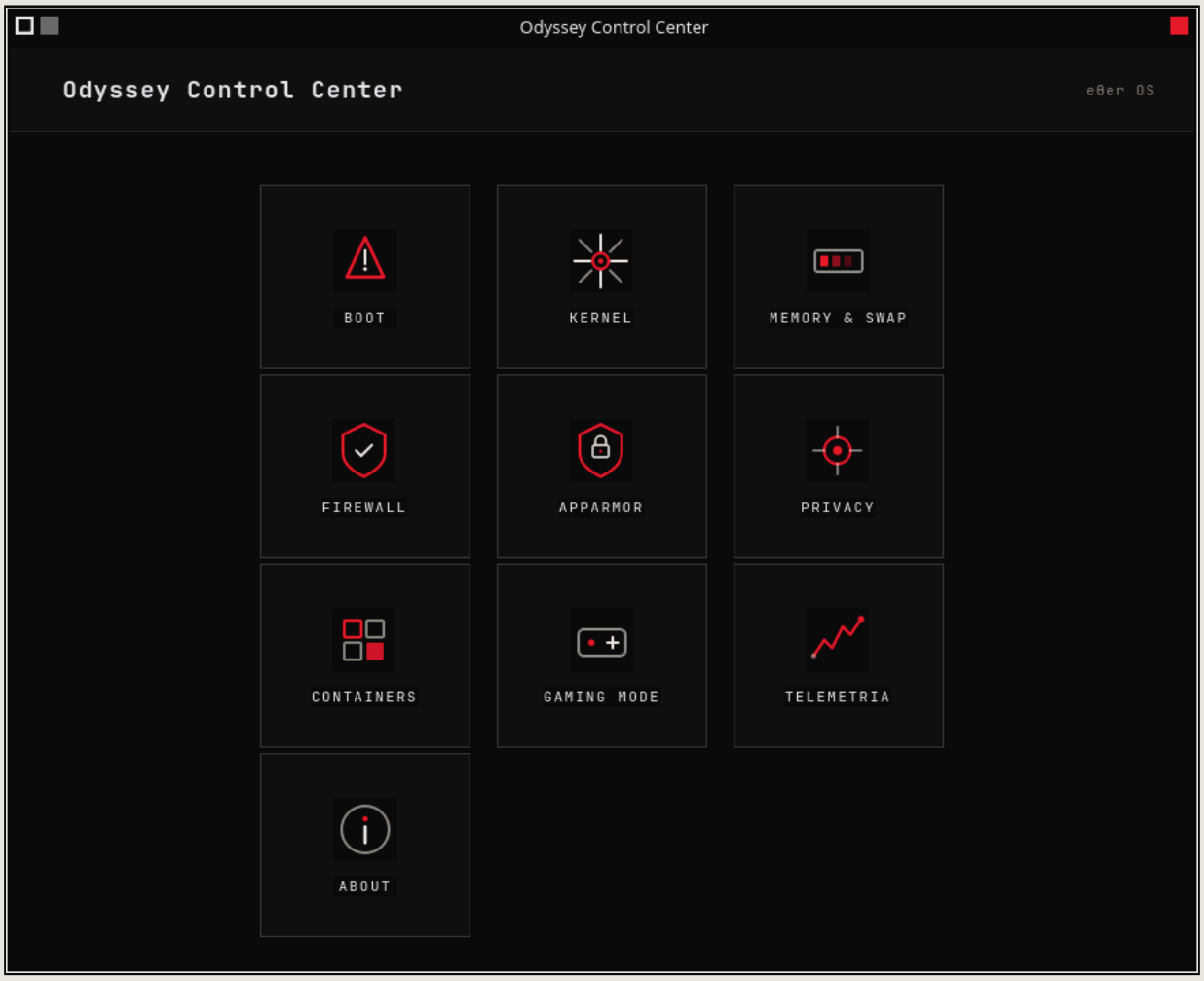

FIG · La schermata iniziale del Control Center — un riquadro per ogni pannello.

Ogni pannello è uno strato grafico sottile sopra il vero meccanismo Linux sottostante: la config di GRUB, l’interfaccia sysctl del kernel stesso, `nftables`, `podman`. Un pannello non inventa mai un proprio stato privato — legge i file e i comandi reali, te li mostra, e scrive esattamente quello che hai chiesto. Niente di ciò che fa è nascosto, e tutto quello che fa si può fare anche a mano, da terminale, con i comandi esatti che questa guida mostra accanto a ogni pulsante.

Questa è una guida pratica, non un elenco di ogni opzione — risponde a “cosa clicco, e cosa succede se lo faccio.” Ogni capitolo ha la stessa struttura in due parti: prima una **guida visiva**, screenshot dopo screenshot, pulsante dopo pulsante, scritta per chi non ha mai aperto il Control Center prima — poi una sezione **ADVANCED**, segnalata da un banner nero, che entra nel dettaglio: percorsi esatti dei file, i comandi reali che ogni pulsante esegue sotto il cofano, e cosa fare da terminale se preferisci saltare

del tutto il pannello. Leggi solo la guida visiva, oppure salta direttamente ad Advanced se conosci già Linux — sono entrambe complete a sé stanti.

Per i due pannelli con implicazioni di sicurezza reali — Firewall e AppArmor — il riferimento approfondito è la **Security and Hardening Guide**; qui trovi la guida visiva, l'essenziale, e un rimando.

1.1 Pattern che ritrovi in ogni pannello

Alcune convenzioni si ripetono in tutti e dieci i pannelli — impararle una volta vale per tutti.

LA RICHIESTA DELLA PASSWORD, UNA SOLA VOLTA PER PANNELLO

La maggior parte dei pannelli deve leggere o scrivere file che appartengono a root — la config di GRUB, sysctl, il ruleset del firewall. La prima volta che un pannello ne ha bisogno, ricevi un solo dialogo di autenticazione. I pannelli sono costruiti per raggruppare le loro letture: aprire Boot, per esempio, legge la config di GRUB **e** lo stato del firmware EFI in un'unica richiesta, così digiti la password una volta sola, non due. Riaprire un pannello nella stessa sessione non richiede la password di nuovo; le azioni successive (applicare una modifica, ricaricare) mostrano il proprio avanzamento nel log del pannello stesso.

IL TRACKER DEGLI STEP

I pannelli che ti guidano attraverso l'installazione di qualcosa prima che diventi utilizzabile — os-prober in Boot, SearXNG e Tor in Privacy, il motore container in Containers — mostrano un piccolo tracker in alto: una riga di step, con quello corrente evidenziato, e un solo pulsante d'azione che fa sempre **lo step successivo**, qualunque esso sia. Non devi mai indovinare cosa cliccare per primo.

IL LOG IN FONDO

Ogni pannello termina in un piccolo log scorrevole. Non è decorazione — è il comando esatto che è stato eseguito, il suo output, e un verdetto colorato (verde per successo, rosso per fallimento). Se un pannello fa qualcosa che non capisci, il log è dove sta la risposta vera.

TRY VS. MAKE PERSISTENT

Una coppia di pulsanti ricorrente, soprattutto in Kernel e Memory: **TRY NOW** (o **APPLY NOW**) cambia il sistema in esecuzione immediatamente, e torna indietro al prossimo riavvio se non fai altro. **MAKE PERSISTENT** scrive lo stesso valore in un file di config sotto `/etc/`, così sopravvive anche ai riavvii. La distinzione esiste apposta — provare un valore non costa nulla, e niente è permanente finché non lo dici tu, due volte.

CHAPTER 2

Boot

Il pannello Boot ha due tab su qualunque macchina che si avvia in modalità UEFI: **GRUB** (il menù di avvio vero e proprio) ed **EFI** (il livello firmware sottostante). Sulle macchine più vecchie, solo BIOS, esiste solo GRUB, perché il livello EFI sotto semplicemente non c'è.

> DUE LIVELLI, NON UNO
>
> **GRUB** è il menù di avvio che vedi sullo schermo — decide quale sistema operativo parte una volta che GRUB stesso è già in esecuzione. Il **firmware EFI/UEFI** è un livello sotto GRUB — decide a quale bootloader la macchina passa il controllo per primo, prima ancora che GRUB si sia caricato. La maggior parte delle persone non tocca mai la tab EFI; esiste per i casi che GRUB non può raggiungere, come una seconda installazione Windows che gestisce la propria voce di boot in modo indipendente.

2.1 Guida visiva — trovare un altro sistema operativo e impostare il default

La tab GRUB si apre sulla sezione **Other OS detection**. Odyssey non scansiona altri sistemi per default — la maggior parte delle installazioni ha solo Odyssey sul disco, e scansionare costa tempo a ogni rigenerazione del menù di avvio per niente. Attivarla è una sequenza guidata di tre step, mostrata come un piccolo tracker numerato.

1. **`Installed`**

   Lo strumento di scansione, os-prober, non è ancora sul sistema. Un solo pulsante: **INSTALL OS-PROBER**.

2. **`Enable scan`**

   Lo strumento è installato, ma GRUB è ancora configurato per ignorarlo — il default di Odyssey. Un solo pulsante: **ENABLE SCAN**.

3. **`Scan`**

   GRUB ora lo userà. Un solo pulsante, e quello che userai di nuovo ogni volta che aggiungi o rimuovi un altro sistema in futuro: **SCAN & REGENERATE**.

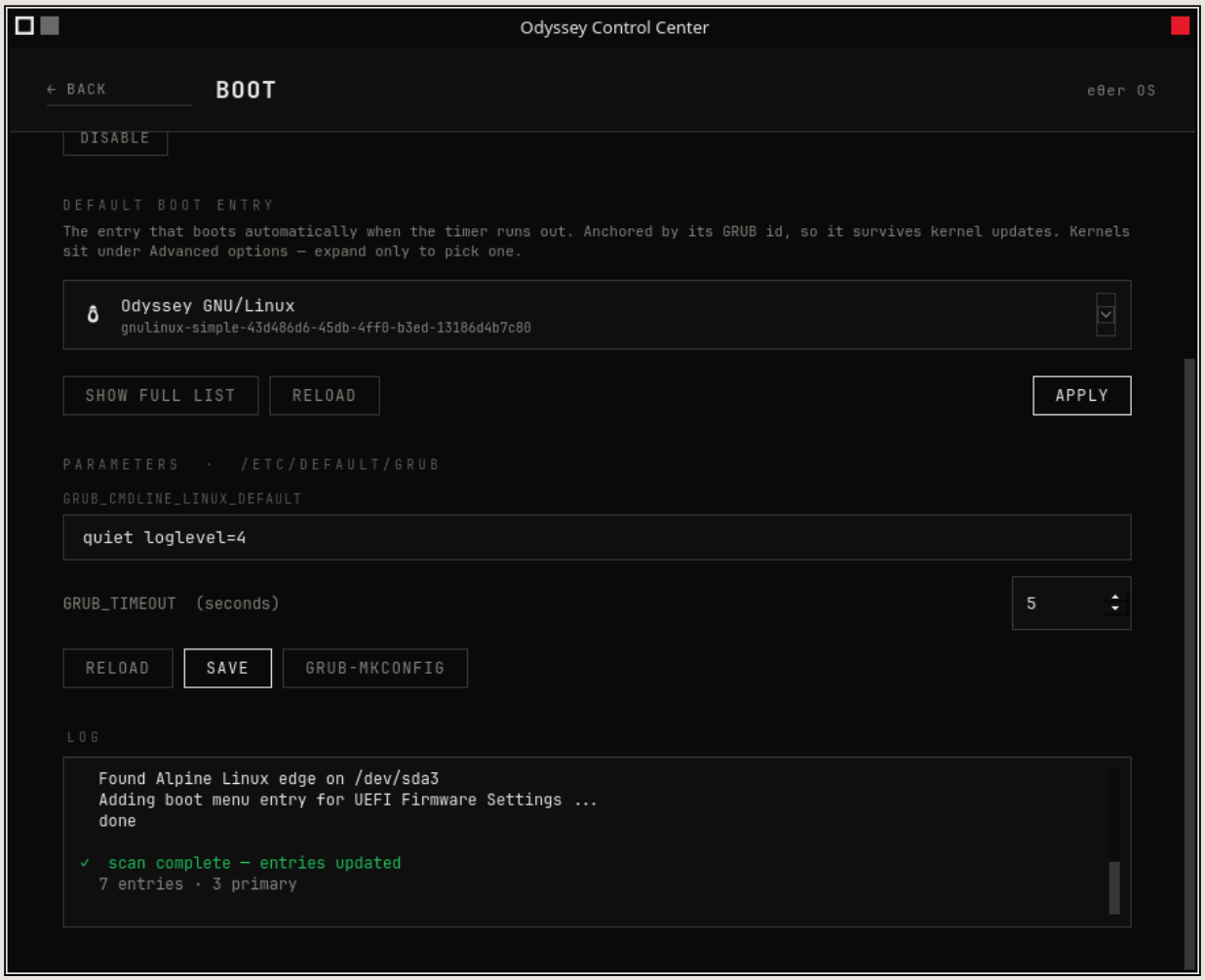

FIG · Tutti e tre gli step verdi — os-prober installato, scansione abilitata, pronto. Il pulsante d'azione unico corrisponde sempre allo step corrente.

Clicca sugli step (ognuno richiede di nuovo l'autenticazione solo se necessario) ed esegui la scansione. Il pannello mostra esattamente cosa ha trovato prima che tu lasci la sezione:

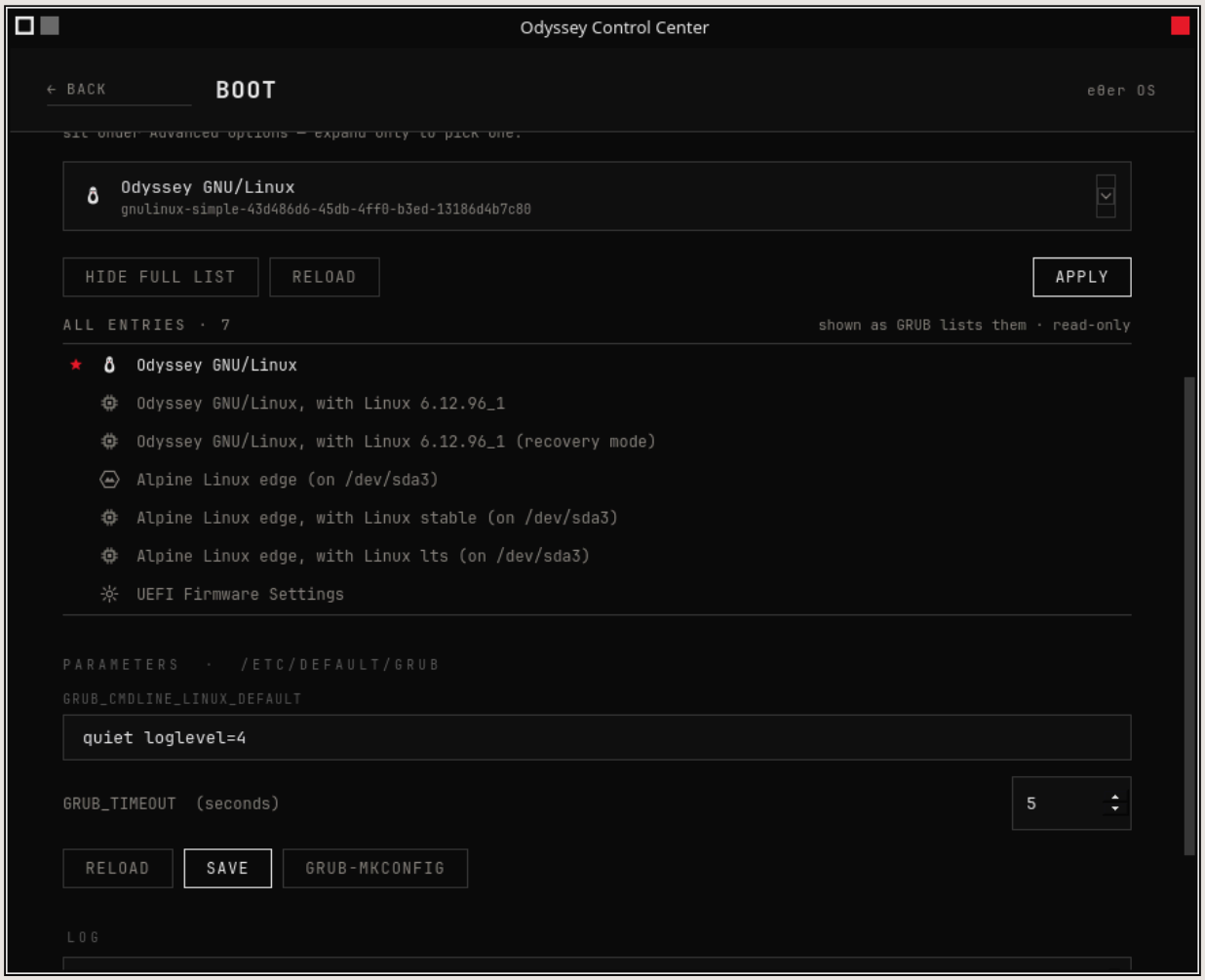

FIG · Scansione completata: Alpine Linux trovato su /dev/sda3, insieme alle voci di Odyssey stesso e alla voce UEFI Firmware Settings — 7 voci, 3 primarie.

Il log lo conferma in termini semplici — **“scan complete — entries updated · 7 entries · 3 primary.”** Qualunque cosa os-prober trovi ora compare automaticamente nella sezione sottostante, pronta per essere scelta.

2.2 Guida visiva — scegliere cosa si avvia per default

Scorri fino a **Default boot entry**. Mostra il default corrente come una riga; cliccandola si apre un menù a tendina con tutto ciò che puoi scegliere al suo posto.

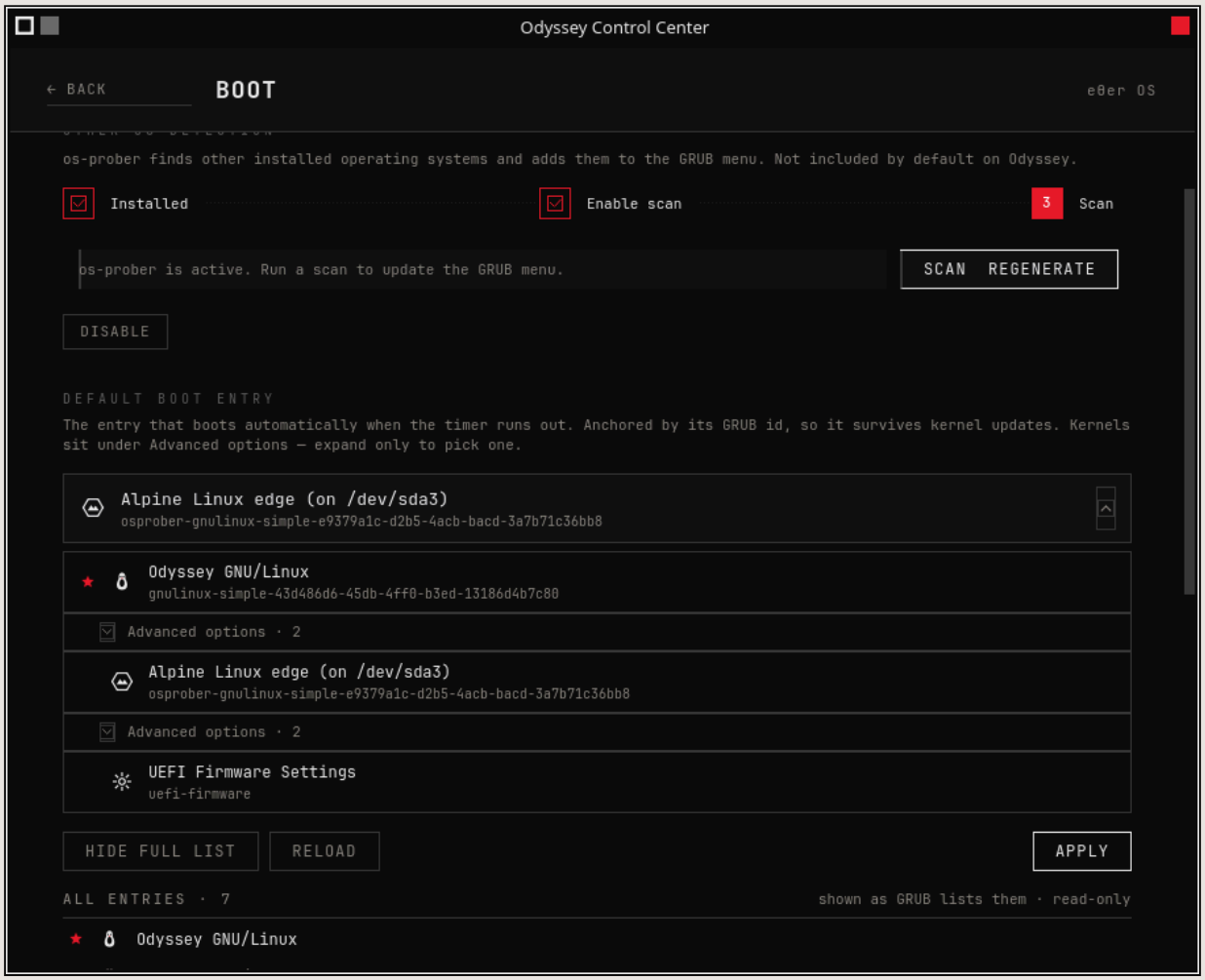

FIG · Il menù a tendina aperto, con Alpine Linux selezionato come scelta in sospeso — non ancora applicata.

Scegliere una voce dalla lista **mette in sospeso** la scelta — non ha ancora effetto. Si vede che è solo in sospeso perché il riquadro mostra ancora la nuova selezione ma niente è stato scritto. Il pulsante che la conferma è **APPLY**, a destra:

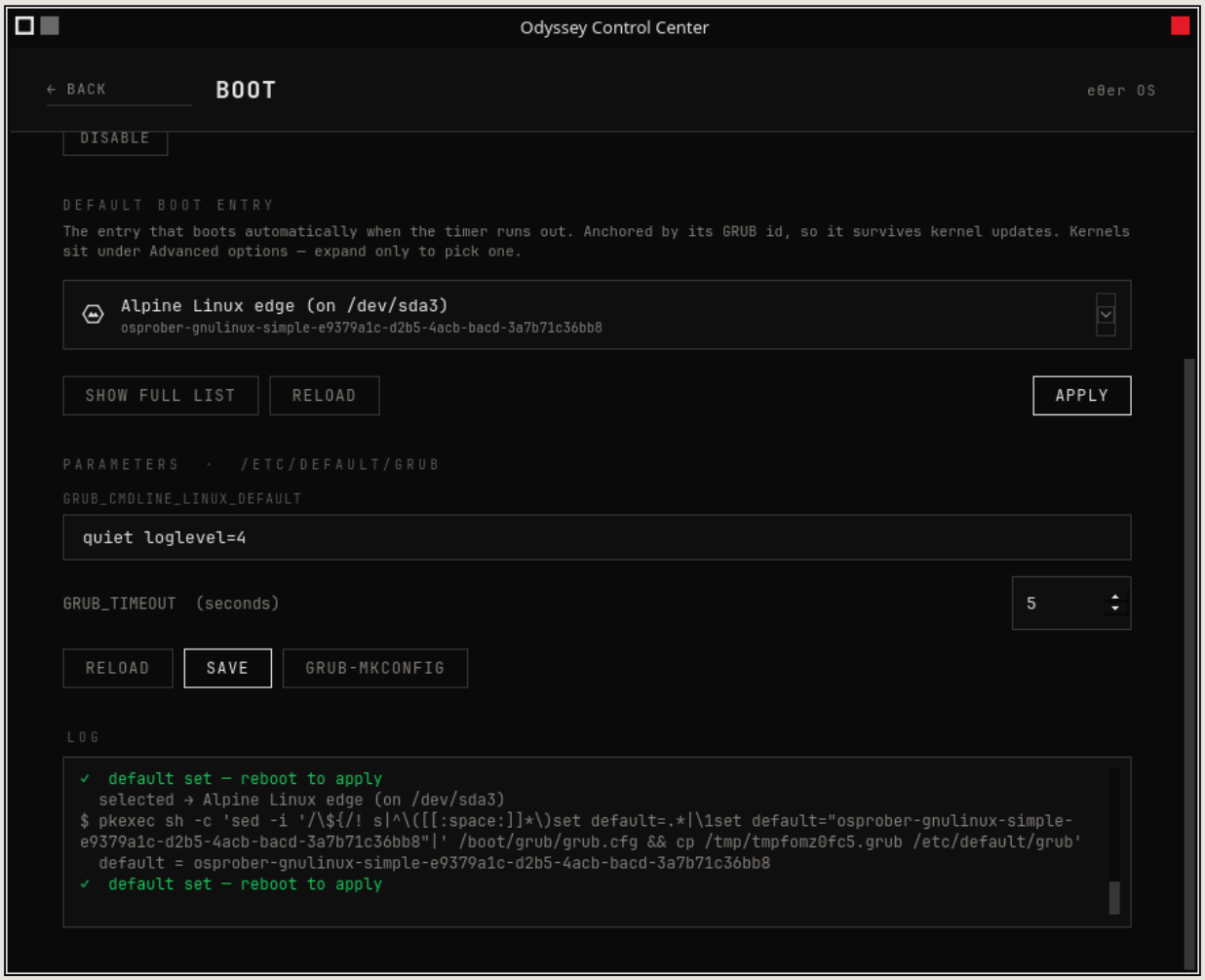

FIG · Dopo APPLY: il log conferma “default set — reboot to apply,” e il selettore ora mostra Alpine Linux come default attivo.

WHAT	Boot → tab GRUB → clicca la voce di default corrente → scegli una voce diversa dal menù a tendina → clicca **APPLY**.
WHY	Fai dual-boot e vuoi che un sistema diverso parta automaticamente, oppure un aggiornamento del kernel ha lasciato selezionato un kernel vecchio e vuoi tornare all’ultimo.
RESULT	La voce scelta si avvia automaticamente dal prossimo riavvio in poi. La modifica è ancorata all’id stabile di quella voce, quindi sopravvive ai futuri aggiornamenti del kernel.
RISK	Nessuno per i tuoi dati — nel peggiore dei casi scegli la voce sbagliata e torni a cambiarla di nuovo.

SHOW FULL LIST mostra ogni voce esattamente come la elenca GRUB stesso — sola lettura, raggruppata in modo che le varianti per singolo kernel stiano sotto una riga espandibile “Advanced options” invece di affollare la lista principale. **RELOAD** rilegge la config di GRUB se pensi sia cambiata al di fuori del pannello.

2.3 Guida visiva — la tab EFI

Clicca la tab **EFI** in alto. Questo è il livello firmware — quello che il menù di avvio della tua scheda madre mostra prima ancora che qualunque sistema operativo sia partito.

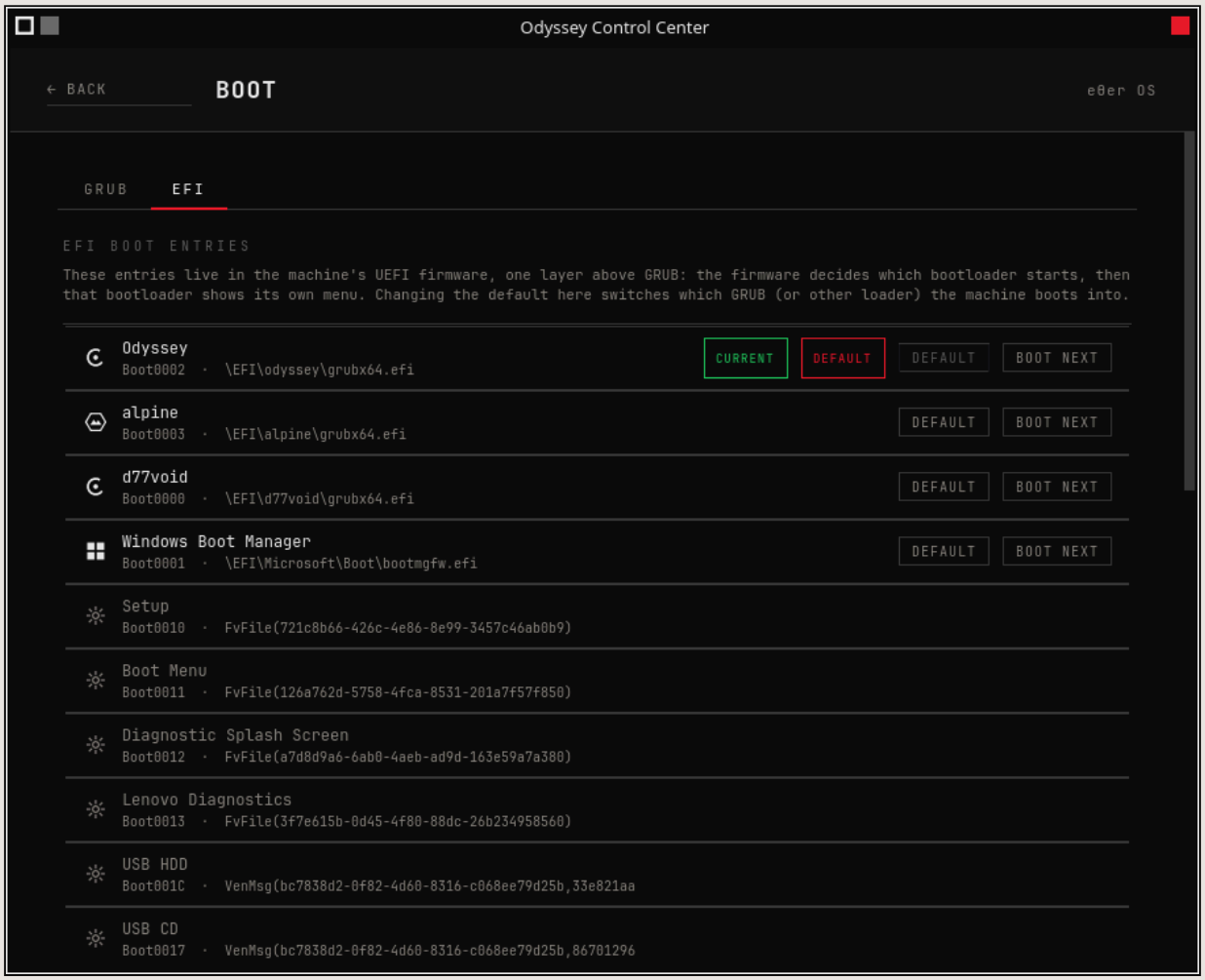

FIG · La tab EFI: Odyssey contrassegnato CURRENT e DEFAULT in verde/rosso, alpine, d77void e Windows Boot Manager ciascuno coi propri pulsanti DEFAULT / BOOT NEXT, più le voci di utilità del firmware sotto.

Ogni riga è una voce del firmware — il suo nome, il suo id GRUB (`Boot0002` e così via), e il file `.efi` esatto a cui punta. Due pulsanti per riga:

PULSANTE	COSA FA
DEFAULT	Rende questa voce il default permanente del firmware — parte a ogni riavvio da questo momento in poi.
BOOT NEXT	Un'eccezione una tantum — questa voce parte solo al **prossimo** riavvio, poi l'ordine normale riprende automaticamente. Un banner appare in alto mentre una scelta è in sospeso, con un pulsante CANCEL.

WHAT	Boot → tab EFI → trova la voce che vuoi (es. "Odyssey" o "Windows Boot Manager") → clicca DEFAULT, oppure BOOT NEXT per un'eccezione una tantum.
WHY	Se il menù di avvio della tua scheda madre sta avviando il bootloader sbagliato — non un'impostazione di GRUB, una del firmware — questo è l'unico posto che lo sistema.
RESULT	DEFAULT: la voce si sposta in cima all'ordine di avvio del firmware in modo permanente. BOOT NEXT: parte una volta sola, poi torna indietro.
RISK	Basso, e sempre recuperabile da questa stessa tab o dalla schermata di setup UEFI della tua scheda madre (di solito Canc o F2 all'accensione) se qualcosa si avvia in modo inatteso.

2.4 Parametri del kernel

L'ultima sezione della tab GRUB è la riga di comando del kernel — modificabile come un normale campo di testo, nessuna sintassi GRUB richiesta — e il timeout del menù in secondi. **SAVE** la scrive in `/etc/default/grub`; **GRUB-MKCONFIG** rigenera il vero menù di avvio da quel file, ed è il passaggio che rende effettivo un parametro salvato al prossimo avvio. Il pannello riapplica sempre la tua scelta di default corrente quando rigenera, quindi questo non azzera mai quale sistema si avvia per default.

```
▸ ADVANCED

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

2.5 Advanced

COSA SUCCEDE REALMENTE

La config GRUB di Odyssey è **piatta** — nessun blocco `submenu`. Ogni kernel è un `menuentry` di primo livello, il che significa che un id stabile (non una posizione numerica) può sempre selezionarlo correttamente, anche tra un aggiornamento del kernel e l'altro. Il raggruppamento "Advanced options" che vedi nel selettore è pura convenienza visiva costruita abbinando lo UUID condiviso tra la voce semplice di un sistema operativo e le sue varianti per kernel — il file di GRUB stesso non ha questo raggruppamento.

Impostare un default scrive in due punti nella stessa azione: il `GRUB_DEFAULT` persistente in `/etc/default/grub`, e — poiché `grub.cfg` potrebbe non essere rigenerato immediatamente — una modifica diretta e chirurgica via `sed` della riga `set default="..."` attiva in `/boot/grub/grub.cfg` stesso, saltando il ramo one-shot `${next_entry}` che usa `grub-reboot`. Ecco perché un default ha effetto al prossimo avvio senza dover forzare prima un `grub-mkconfig` completo.

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

OS-PROBER A MANO

```
$ sudo xbps-install -y os-prober
# abilita la scansione — modifica 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 A MANO — EFIBOOTMGR

La tab EFI è un front-end per `efibootmgr`. Tutto quello che fa corrisponde a un comando:

```
# elenca tutte le voci del firmware, nell'ordine di avvio attuale
$ sudo efibootmgr -v
# riordina: metti la voce 0002 per prima (= DEFAULT)
$ sudo efibootmgr -o 0002,0000,0001,0003
# avvio una tantum nella voce 0003 (= BOOT NEXT)
$ sudo efibootmgr -n 0003
# annulla un BootNext in sospeso
$ sudo efibootmgr -N
```

LOADER ORFANI

Occasionalmente un installer scrive il proprio bootloader sulla EFI System Partition senza registrare una voce firmware per esso — il loader esiste sul disco ma il firmware non sa che c'è. Il pannello di Odyssey scansiona direttamente la ESP e li elenca separatamente come **"ON DISK, NOT IN FIRMWARE,"** ciascuno con un REGISTER a un click. A mano, l'equivalente è `efibootmgr -c -d /dev/sdX -p N -L "Nome" -l '\EFI\nome\bootx64.efi'` — nota che le voci appena create vengono messe **per prime** nell'ordine di avvio dal firmware stesso, quindi riapplica DEFAULT sulla tua voce abituale dopo, se non vuoi che prenda il sopravvento.

FILE TOCCATI DA QUESTO PANNELLO

FILE	RUOLO
`/etc/default/grub`	Riga di comando del kernel, timeout, GRUB_DEFAULT, GRUB_DISABLE_OS_PROBER
`/boot/grub/grub.cfg`	Il menù di avvio generato vero e proprio — non modificarlo mai a mano; viene sovrascritto da grub-mkconfig
Variabili EFI (NVRAM)	Voci di avvio del firmware — non è affatto un file; letto e scritto interamente tramite efibootmgr

CHAPTER 3

Kernel

Quattro tab — **SYSCTL**, **CPU GOVERNOR**, **HUGEPAGES**, **MODULES** — più una riga di preset a un click sopra tutte e quattro, che si applica a tutte insieme.

3.1 Guida visiva — i preset

In cima: la versione del tuo kernel in esecuzione, lo scheduler attivo, e quattro pulsanti preset — **DESKTOP**, **GAMING**, **SERVER**, **BALANCED**.

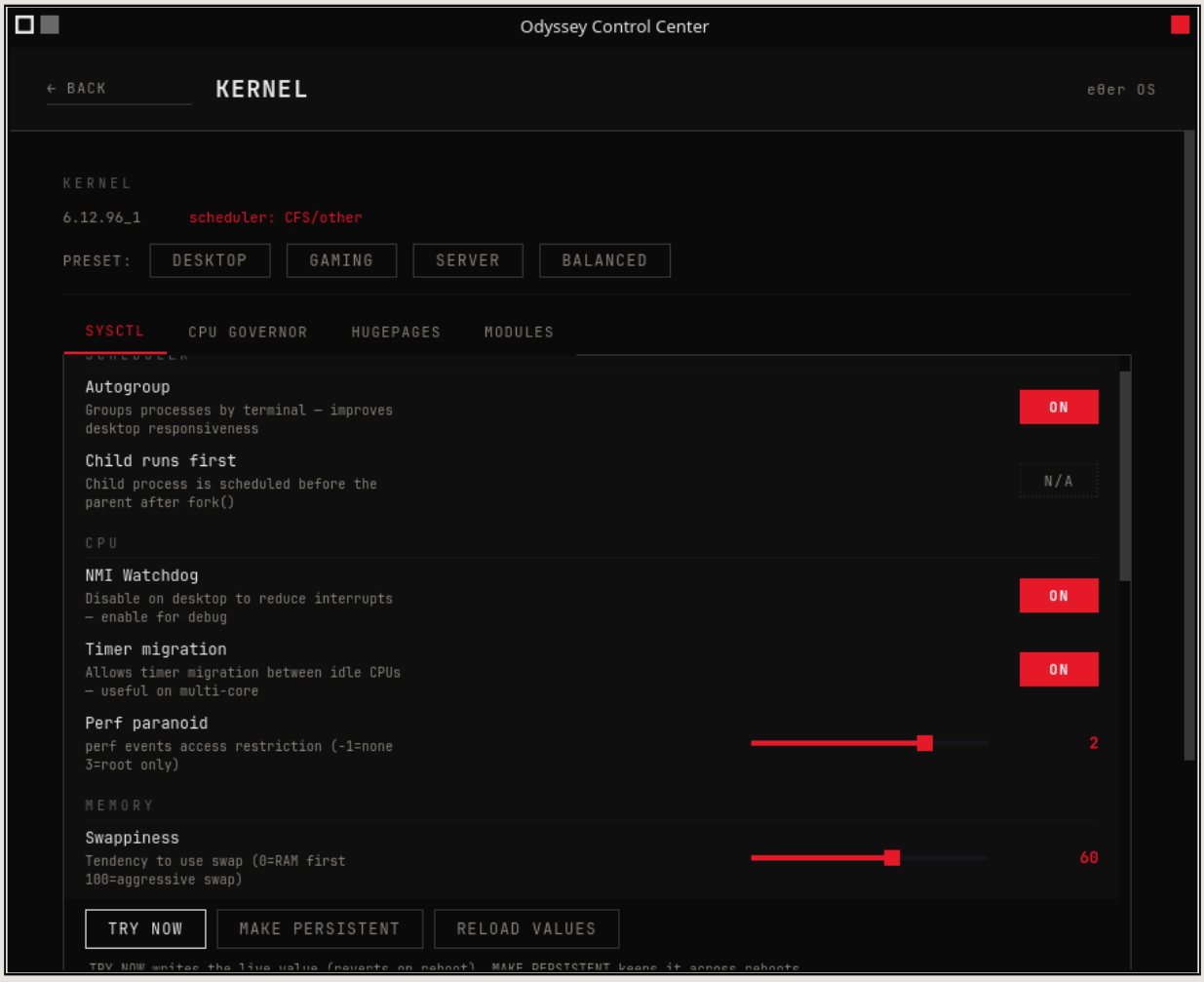

FIG · La tab SYSCTL appena aperta: kernel 6.12.96_1, scheduler CFS/other, i quattro preset, e i gruppi Scheduler/CPU/Memory sotto.

PRESET	QUANDO VALE LA PENA USARLO
DESKTOP	Una macchina di tutti i giorni — navigazione, lavoro d'ufficio, multitasking leggero. Swappiness più bassa, pressione moderata sulla cache, timer migration attiva per un uso morbido dell'energia da fermo. Il default sensato per la maggior parte delle persone.
GAMING	Swappiness spinta quasi a zero (tieni tutto in RAM), timer migration disattivata (i core restano dedicati invece di migrare il lavoro), restrizioni sugli eventi perf allentate per strumenti di profiling come MangoHud. Usalo quando i frame time contano più del consumo energetico da fermo.
SERVER	Swappiness e tolleranza alla pressione della cache più alte, NMI watchdog attivo, restrizioni più severe su puntatori kernel/log. Usalo su una macchina headless che privilegia stabilità e sicurezza rispetto alla reattività interattiva.
BALANCED	Una via di mezzo tra Desktop e Gaming — per una macchina che a volte è una workstation, a volte una sessione di gioco, e non vuoi passare da un preset all'altro.

WHAT	Kernel → clicca DESKTOP, GAMING, SERVER o BALANCED.
WHY	Regolare a mano una dozzina di valori sysctl è tanto chiedere a chi vuole solo "buone impostazioni per giocare" — un preset è esattamente questo, già deciso.
RESULT	Ogni valore nella tab SYSCTL salta ai numeri del preset, predisposto e visibile — devi comunque TRY NOW o MAKE PERSISTENT (sotto) per applicarlo davvero.
RISK	Nessuno — scegliere un preset compila solo il modulo.

3.2 Guida visiva — SYSCTL, gruppo per gruppo

Scorri in basso e ogni parametro è raggruppato sotto un'intestazione, ognuno con una descrizione in linguaggio semplice direttamente nel pannello — non devi mai sapere cosa significa `vm.compaction_proactiveness` senza prima leggerlo lì.

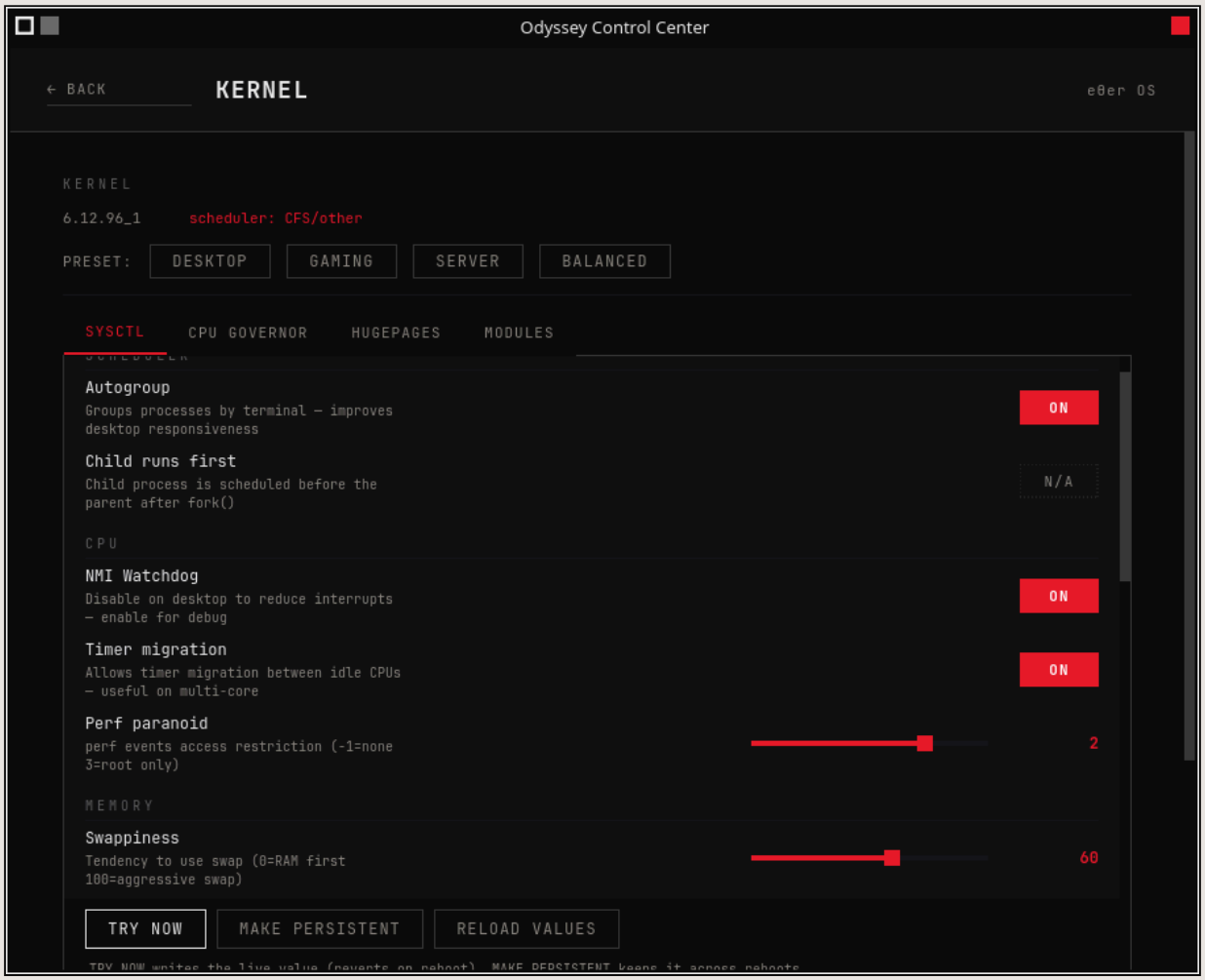

FIG · Gruppi Scheduler e CPU: Autogroup, Child runs first, NMI Watchdog, Timer migration, Perf paranoid — ciascuno col proprio interruttore o slider e una riga di spiegazione.

GRUPPO	COSA CONTIENE
Scheduler	Come i processi vengono raggruppati e prioritizzati
CPU	Watchdog, timer migration, restrizioni sul monitoraggio delle prestazioni
Memory	Swappiness, pressione della cache, soglie di scrittura delle pagine sporche
Network	TCP Fast Open, dimensione della coda di backlog, limiti dei buffer socket
Security	Se puntatori kernel e messaggi di log sono nascosti agli utenti senza privilegi

Due pulsanti in fondo applicano le tue modifiche — e sono deliberatamente diversi:

WHAT	TRY NOW
WHY	Testare un valore con zero rischio che sopravviva a un riavvio andato male.
RESULT	Scritto live via `sysctl` — attivo immediatamente, solo per questa sessione.
RISK	Nessuno — torna indietro automaticamente al riavvio.

WHAT	MAKE PERSISTENT
WHY	Hai provato un valore (o un intero preset) e vuoi che sopravviva ai riavvii.
RESULT	Scritto in `/etc/sysctl.d/99-odyssey.conf` **e** applicato live nello stesso click.
RISK	Basso — è un file di testo; cancella la riga o riapplica per annullare.

RELOAD VALUES rilegge cosa è attualmente attivo, scartando qualunque cosa digitata ma non applicata.

3.3 Guida visiva — CPU governor

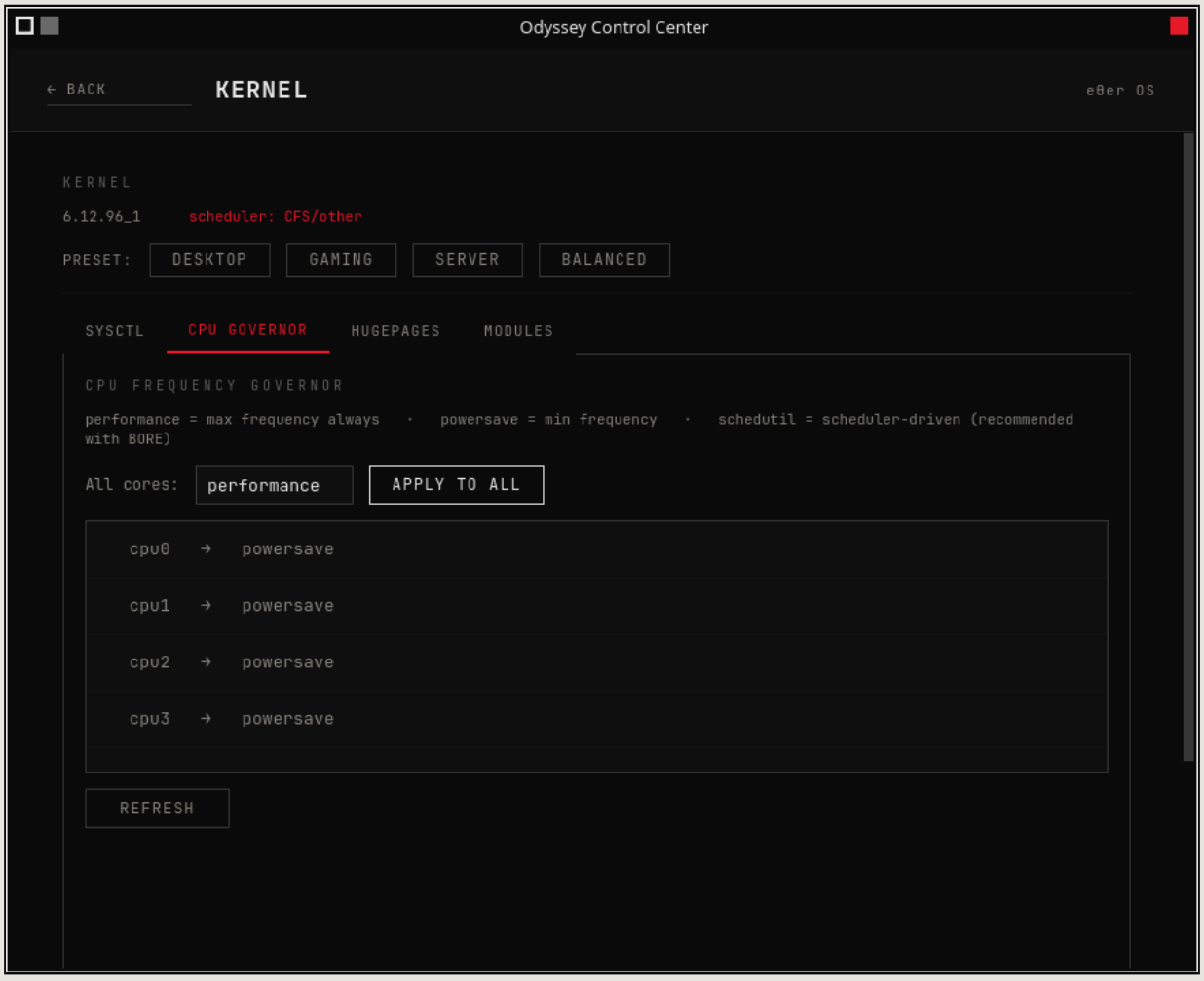

FIG · La tab CPU GOVERNOR: “All cores” impostato su performance con APPLY TO ALL, e una lista per singolo core sotto che mostra ancora tutti powersave — una discrepanza in attesa di essere corretta.

Il governor decide quanto aggressivamente la tua CPU cambia frequenza di clock. Un menù a tendina applica un governor a tutti i core in una volta; la lista sotto mostra il governor **live** per ogni singolo core — è così che scopri un core bloccato sull’impostazione sbagliata, esattamente come nello screenshot sopra dove “All cores” dice performance ma ogni core sotto legge ancora powersave, il che significa che APPLY TO ALL non è ancora stato cliccato.

GOVERNOR	COMPORTAMENTO
performance	Frequenza massima, sempre — più reattivo, più consumo
powersave	Frequenza minima, sempre — più silenzioso, meno reattivo
schedutil	Frequenza guidata dalla visione del carico dello scheduler stesso — consigliato con lo scheduler BORE

Solo i governor che la tua specifica CPU e kernel espongono davvero compaiono nel menù a tendina.

3.4 Guida visiva — hugepages

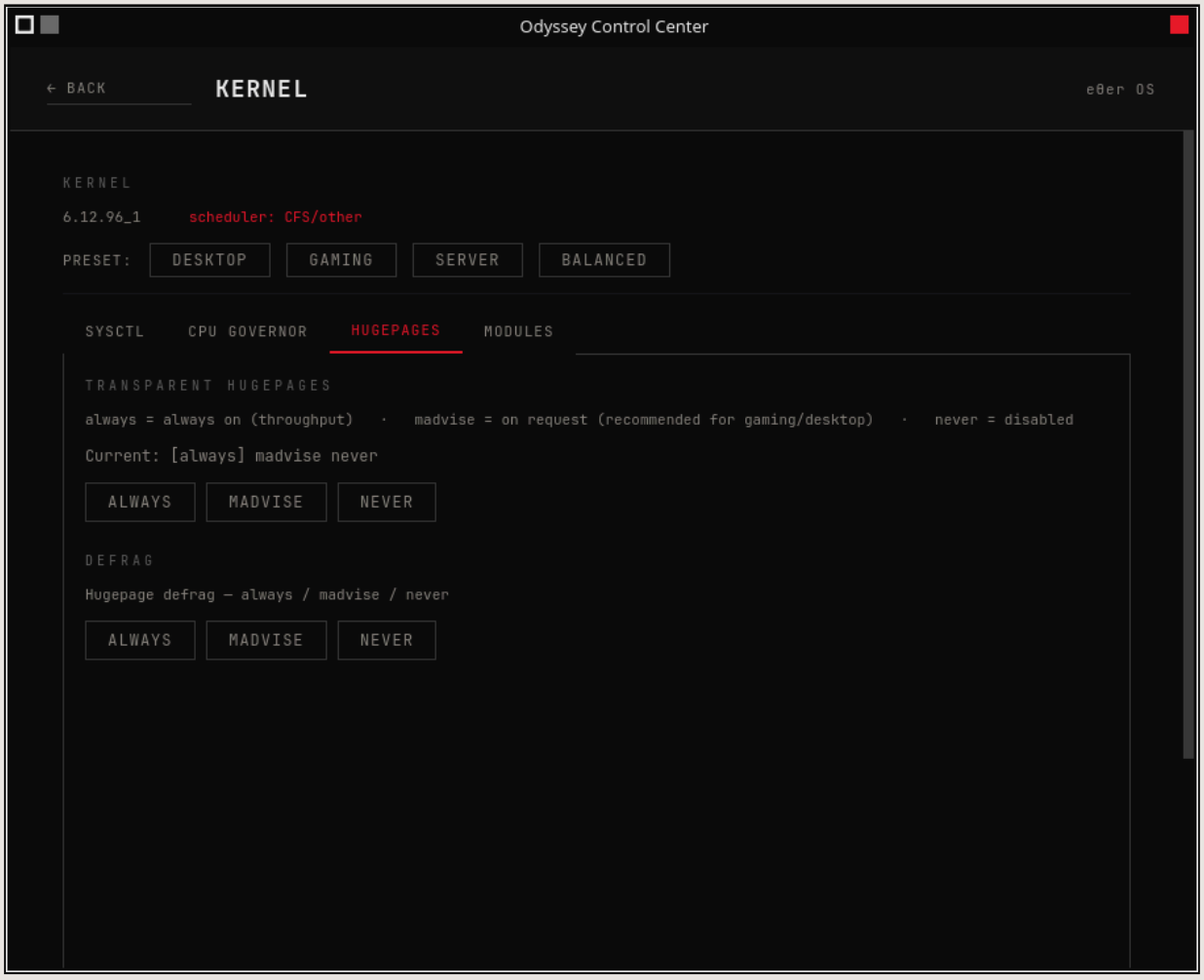

FIG · La tab HUGEPAGES: Transparent Hugepages che mostra “Current: [always] madvise never” con ALWAYS evidenziato, e la stessa riga di tre pulsanti di nuovo per Defrag sotto.

Due impostazioni indipendenti, ciascuna una riga di tre pulsanti applicati all’istante del click — il valore live corrente è sempre mostrato tra parentesi sopra i pulsanti, come `[always] madvise never` nello screenshot.

IMPOSTAZIONE	COSA CONTROLLA
Transparent hugepages	ALWAYS = sempre attivo ovunque (massimo throughput); MADVISE = solo quando un’app lo richiede esplicitamente (consigliato per gaming/desktop); NEVER = completamente disattivo

IMPOSTAZIONE	COSA CONTROLLA
Defrag	Quanto aggressivamente il kernel compatta la memoria per **formare** hugepages su richiesta — stesse tre opzioni, indipendente dall'impostazione sopra

3.5 Guida visiva — modules

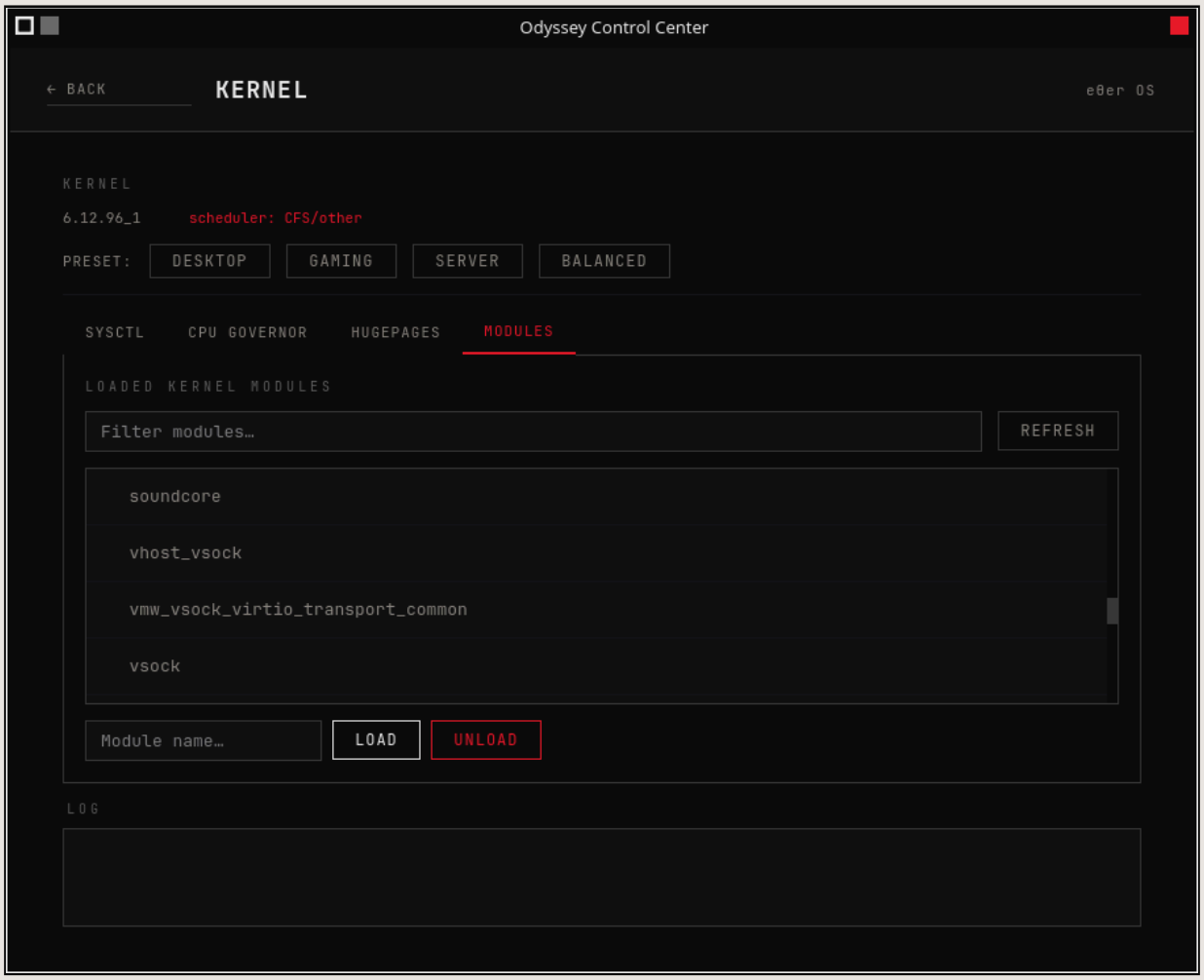

FIG · La tab MODULES: un campo filtro, una lista live (soundcore, vhost_vsock, vsock…), e una riga LOAD/UNLOAD in fondo per un modulo digitato per nome.

Una lista live e filtrabile di ogni modulo del kernel attualmente caricato — digita nel filtro per restringerla, REFRESH per rileggerla. A differenza delle altre tre tab, questa ti permette di agire direttamente: digita un nome di modulo e clicca **LOAD** o **UNLOAD**.

WHAT	Kernel → tab MODULES → digita un nome di modulo → LOAD o UNLOAD.
WHY	Occasionalmente serve un driver o modulo specifico attivo (o rimosso) senza riavviare — testare hardware, liberare un dispositivo che un altro modulo sta trattenendo.
RESULT	Il modulo viene inserito o rimosso dal kernel in esecuzione immediatamente.
RISK	Moderato specificamente per UNLOAD — rimuovere un modulo da cui dipende qualcos'altro può destabilizzare ciò che lo stava usando (un driver video, un filesystem). LOAD è generalmente sicuro; il kernel rifiuta subito di caricare un modulo incompatibile invece di applicarlo a metà.

▸ **ADVANCED**

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

3.6 Advanced

LE CHIAVI SYSCTL ESATTE

CHIAVE	ETICHETTA NEL PANNELLO
`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

VALORI DEI PRESET, 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
# le tabelle complete dei preset si leggono direttamente da panel_kernel.py —
# apri il sorgente del Control Center se vuoi ogni singolo campo
```

A MANO, SENZA IL PANNELLO

```
# live, temporaneo
$ sudo sysctl vm.swappiness=10
# persistente — lo stesso file che scrive il pannello
$ echo “vm.swappiness=10” | sudo tee -a /etc/sysctl.d/99-odyssey.conf
$ sudo sysctl -system
# CPU governor, per core
$ 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
# moduli
$ sudo modprobe MODULE_NAME
$ sudo modprobe -r MODULE_NAME
```

PERCHÉ TRY NOW E MAKE PERSISTENT SONO CHIAMATE SEPARATE

Le scritture `sysctl` sono per natura sempre solo live — il kernel non ha una persistenza integrata. MAKE PERSISTENT è Odyssey che sceglie un file canonico unico, `/etc/sysctl.d/99-odyssey.conf`, e fa sia la scrittura live che quella su file insieme in modo che i due non si disallineino mai. Modificare quel file direttamente ed eseguire `sysctl --system` dopo è funzionalmente identico a cliccare il pulsante.

CHAPTER 4

Memory & Swap

Una sola schermata, tre sezioni: un riepilogo in sola lettura, swappiness, e ZRAM.

4.1 Guida visiva

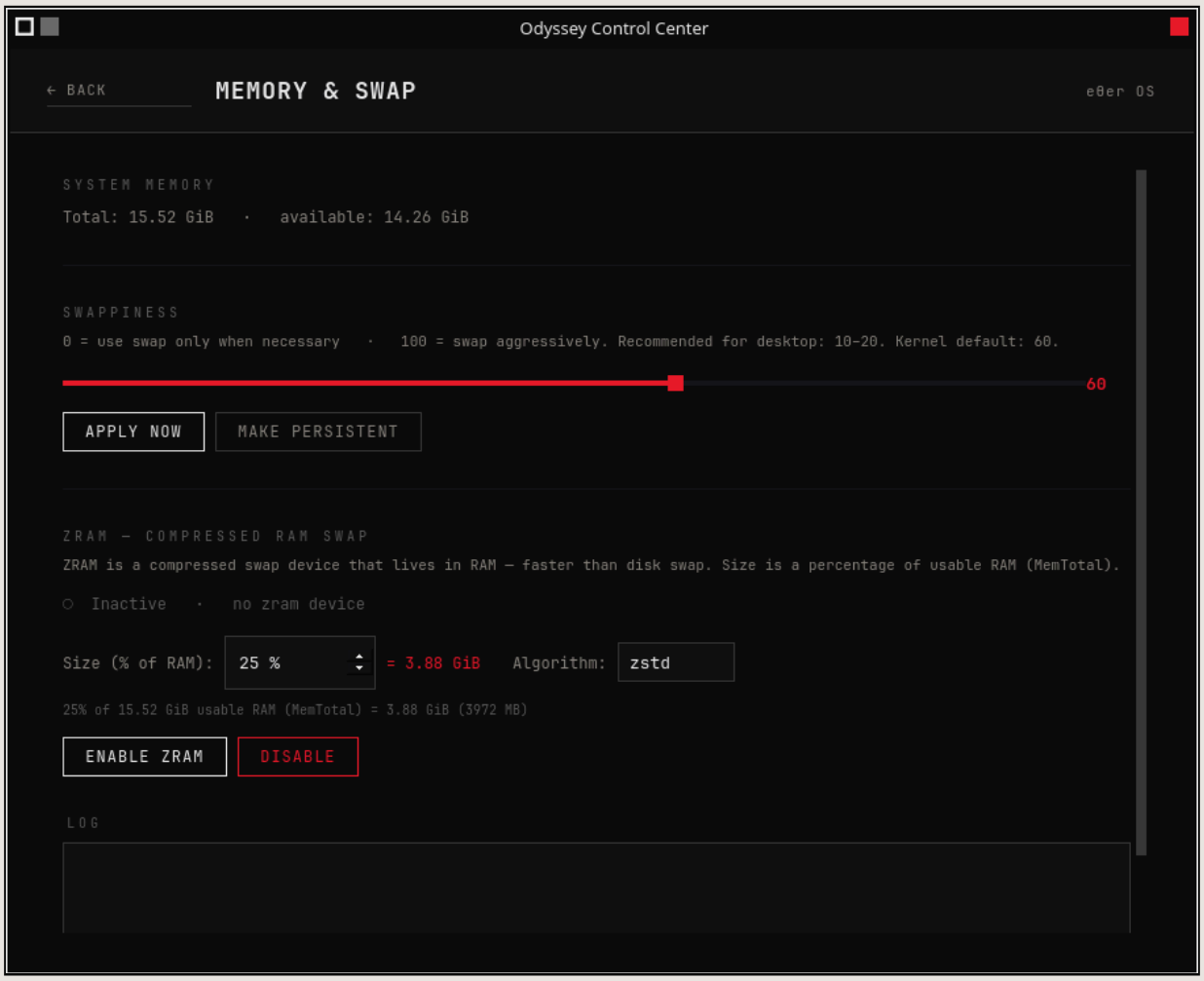

FIG · Il pannello completo: System memory (15,52 GiB totali, 14,26 GiB disponibili), lo slider Swappiness a 60 con APPLY NOW / MAKE PERSISTENT, e ZRAM sotto che mostra “Inactive · no zram device” con dimensione, algoritmo ed ENABLE/DISABLE.

SYSTEM MEMORY

Solo un riepilogo in cima — RAM totale e cosa è attualmente disponibile. Nessun controllo; è il contesto per le due sezioni sotto.

SWAPPINESS

Un singolo slider da 0 a 100. 0 significa “swap solo quando non c’è davvero altra scelta”; 100 significa “swap aggressivo, ben prima che la RAM sia piena.” Il default del kernel è 60 (quello che mostra lo screenshot); il suggerimento del pannello stesso indica 10–20 per un desktop — lo stesso numero che imposta il preset DESKTOP del pannello Kernel.

WHAT	Trascina lo slider → APPLY NOW per provare, MAKE PERSISTENT una volta che sei soddisfatto.
WHY	Un valore più basso mantiene ciò che usi attivamente nella RAM veloce invece di spostarlo nello swap più lento — di solito quello che vuoi con abbastanza memoria installata.
RESULT	APPLY NOW lo cambia live, immediatamente, solo per questa sessione. MAKE PERSISTENT lo scrive in un file di config così sopravvive anche al riavvio.
RISK	Molto basso. Un valore estremamente basso su una macchina davvero a corto di RAM può renderla più incline a esaurire la memoria bruscamente sotto carico pesante invece di fare swap in modo graduale — se succede, alzalo di nuovo.

ZRAM — SWAP COMPRESSO IN RAM

ZRAM è un dispositivo di swap che vive **dentro** la RAM stessa, compresso — decisamente più veloce dello swap su disco perché nessun disco è coinvolto. La dimensione è una percentuale della tua RAM utilizzabile totale, non un numero fisso, così l’impostazione si adatta sensatamente a macchine diverse; un valore live accanto al campo dimensione converte la percentuale in GiB per te (nello screenshot: 25%

CHAPTER 5

3,88 GiB su un sistema da 15,52 GiB).

WHAT	Imposta la percentuale di dimensione e scegli un algoritmo (zstd, lz4, lzo, o deflate) → ENABLE ZRAM.
WHY	Lo swap compresso in RAM assorbe la pressione sulla memoria con un impatto sulle prestazioni molto minore rispetto allo swap su disco — utile su qualunque macchina, specialmente una con RAM limitata.
RESULT	Un dispositivo `zram0` viene creato e attivato alla dimensione e algoritmo scelti; la riga di stato passa da "Inactive" ad attivo.
RISK	Basso. zstd (il default) bilancia bene rapporto di compressione e costo CPU per la maggior parte delle macchine. DISABLE lo disattiva in modo pulito in qualunque momento.

▸ **ADVANCED**

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

5.1 Advanced

COSA SIGNIFICA DAVVERO "DIPENDE DALL'HARDWARE" QUI

Entrambe le impostazioni interagiscono col tuo hardware specifico, non solo con "più RAM = meglio":

- **SSD NVMe veloce**: una swappiness nell'intervallo 10–20 costa molto poco anche se fa swap — il disco è abbastanza veloce da renderlo quasi impercettibile. ZRAM vince comunque su latenza e usura dell'SSD, ma il divario è più piccolo rispetto a un disco meccanico.
- **HDD meccanico o eMMC come swap**: tieni la swappiness bassa (0–10) — lo swap su disco su supporti lenti causa lo scatto che si chiama "thrashing." ZRAM è una vittoria molto più grande proprio qui.
- **Margine CPU**: ZRAM scambia I/O su disco con cicli CPU spesi a comprimere e decomprimere. Su una macchina limitata dalla CPU (hardware più vecchio, o una già al limite durante il gaming), `lz4` comprime più veloce di `zstd` con un rapporto peggiore — un compromesso ragionevole se il tempo CPU è la risorsa più stretta. Su qualunque cosa moderna, il rapporto migliore di `zstd` vale il suo costo leggermente più alto.
- **Margine RAM**: una percentuale ZRAM grande (tipo 50%) su una macchina con solo 8 GiB totali lascia meno RAM reale per le applicazioni, poiché ZRAM stessa consuma memoria reale per memorizzare le pagine compresse. Su 16 GiB o più, 25% è un default comodo con margine di riserva.

A MANO

```
# swappiness, live
$ sudo sysctl vm.swappiness=15
# swappiness, persistente — lo stesso file che scrive il pannello
$ echo “vm.swappiness=15” | sudo tee /etc/sysctl.d/99-odyssey.conf
# zram, configurazione manuale
$ 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
# verifica che sia attivo
$ swapon -show
```

PERCHÉ IL PANNELLO RESETTA ZRAM0 INVECE DI RIDIMENSIONARLO LIVE

La dimensione di un dispositivo ZRAM non può essere cambiata mentre è attivo e in uso come swap — il kernel richiede che venga prima disattivato e resettato. Il pulsante ENABLE ZRAM del pannello esegue questa sequenza automaticamente (swapoff, reset, riconfigurazione, swapon), motivo per cui cambiare dimensione o algoritmo su un dispositivo già attivo si presenta comunque come un’unica azione pulita invece che come un errore.

CHAPTER 6

Firewall

Il pannello Firewall ha un proprio capitolo dedicato, a piena profondità, nella **Security and Hardening Guide** — il ruleset spiegato riga per riga, tutti e 21 i preset coi loro rischi individuali, l'editor della config raw, e gli equivalenti da riga di comando. Questo capitolo è la versione guidata visivamente.

6.1 Guida visiva

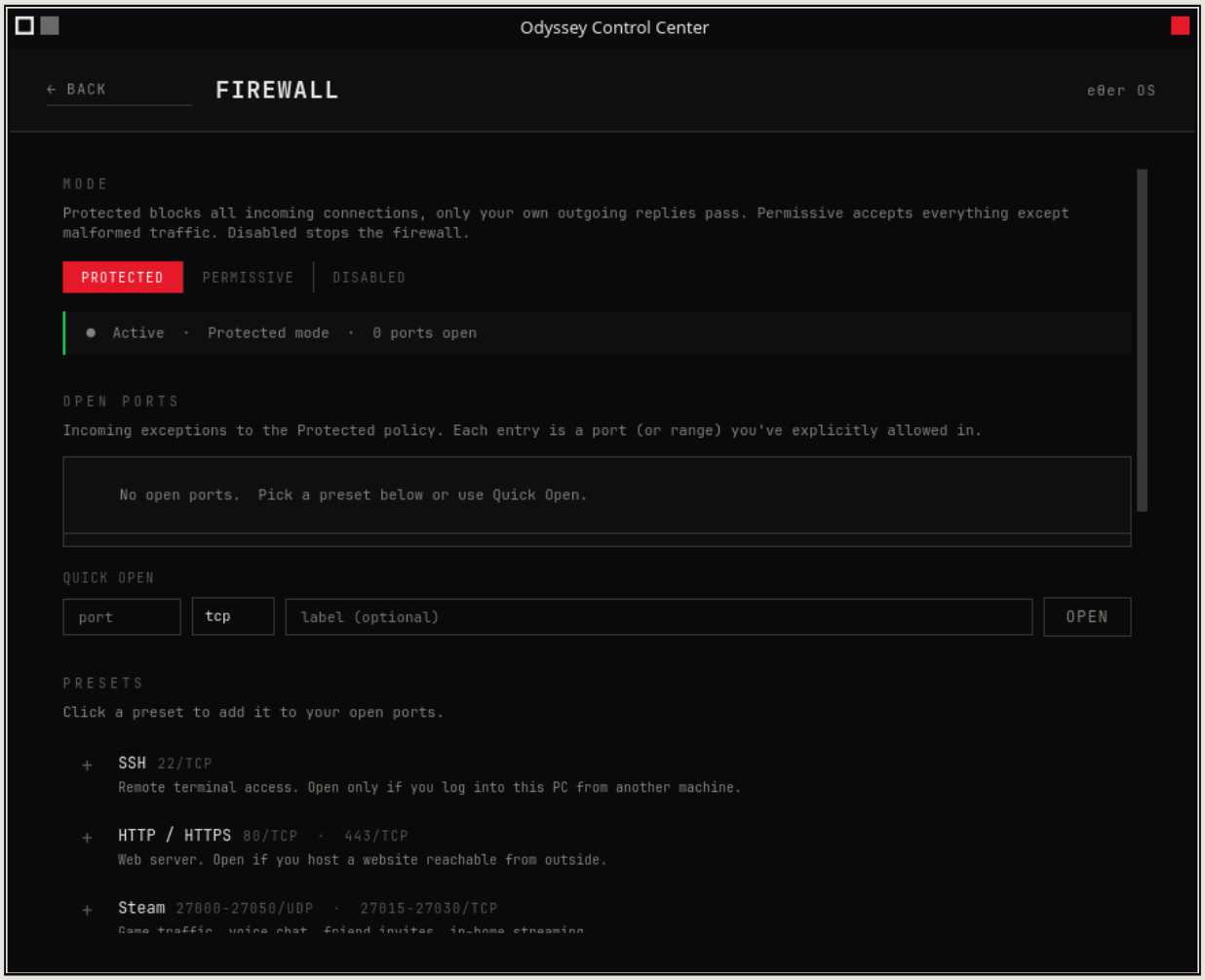

FIG · Modalità PROTECTED attiva, "0 ports open," i campi Quick Open, e i riquadri preset che iniziano con SSH, HTTP/HTTPS e Steam.

MODE

Tre pulsanti in cima, e il pannello spiega ciascuno in una sola riga proprio sopra di essi:

MODALITÀ	COSA FA
PROTECTED	Il default. Blocca tutte le connessioni in entrata; passano solo le tue risposte in uscita.
PERMISSIVE	Accetta tutto tranne il traffico malformato — un'impostazione diagnostica per "è il firewall a rompere questa cosa?", non un posto dove vivere.
DISABLED	Ferma il firewall del tutto.

La riga di stato sotto conferma cosa è realmente attivo — nello screenshot, **"Active · Protected mode · 0 ports open."**

OPEN PORTS E PRESET

OPEN PORTS elenca le tue eccezioni esplicite — vuoto per default, come mostrato. **QUICK OPEN** prende porta, protocollo ed etichetta opzionale direttamente, per qualunque cosa non sia nella lista preset. **PRESETS** sotto offre otto servizi comuni a un click ciascuno, ognuno con una riga che spiega quando ti servirebbe davvero (SSH: "accesso remoto al terminale, apri solo se accedi a questo PC da un'altra macchina"), più altri tredici sotto un "MORE PRESETS" espandibile.

WHAT	Firewall → clicca un riquadro preset, oppure compila QUICK OPEN → SAVE & APPLY.
WHY	Stai eseguendo qualcosa — un server di gioco, un media server, un server di sviluppo — che un'altra macchina deve raggiungere.
RESULT	Quella porta si apre; tutto il resto resta chiuso dalla policy PROTECTED.
RISK	Dipende interamente da quale preset — consulta le note sul rischio per singolo preset nella Security and Hardening Guide prima di aprire qualunque cosa oltre SSH o HTTP/HTTPS su una macchina raggiungibile da internet.

Niente ha effetto finché non premi **SAVE & APPLY** — le modifiche vengono prima messe in sospeso, così puoi rivedere esattamente cosa stai per aprire prima di confermarlo.

```
▸ ADVANCED

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

6.2 Advanced

Il riferimento tecnico completo — il ruleset di default `nftables.conf` spiegato regola per regola, ognuno dei 21 preset con la sua esposizione specifica e la mitigazione, ricette per limitare l'accesso alla sola LAN, gli equivalenti da riga di comando `nft`, e la risoluzione di una sessione SSH bloccata fuori — vive nel **Capitolo 2 della Security and Hardening Guide** (`/usr/share/doc/odyssey/` sul sistema installato). Questo manuale non lo duplica deliberatamente; considera quel capitolo la continuazione di questa sezione.

```
# il comando che vale la pena conoscere subito:
$ sudo nft list ruleset
# mostra esattamente cosa è caricato nel kernel, indipendentemente dal pannello
```

CHAPTER 7

AppArmor

Cinque tab: **STATUS**, **PROFILES**, **COMMUNITY**, **VIOLATIONS**, **EDITOR**. Stesso rimando di Firewall: il riferimento approfondito — cos'è davvero il controllo d'accesso obbligatorio, il flusso completo di confinamento, un esempio pratico — vive nella Security and Hardening Guide. Questa è la guida visiva.

7.1 Guida visiva — STATUS

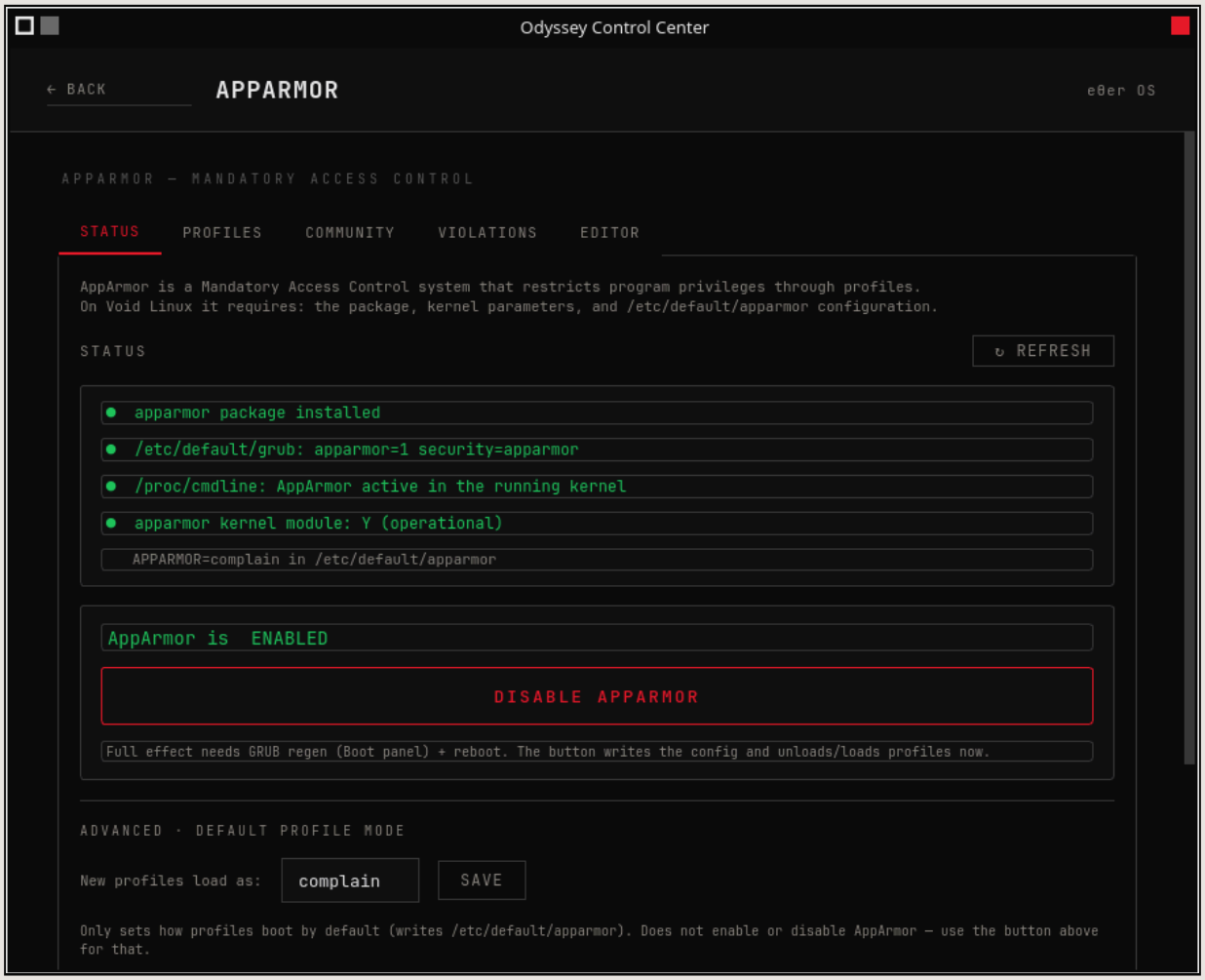

FIG · Tab STATUS: quattro spunte verdi (pacchetto installato, parametri GRUB, kernel in esecuzione, modulo kernel), "AppArmor is ENABLED" con un pulsante rosso DISABLE APPARMOR, e la modalità di default dei profili impostata su complain.

Quattro controlli verificano che il framework funzioni davvero, dall'alto in basso: il pacchetto `apparmor` è installato, i parametri kernel di GRUB sono impostati, il kernel **in esecuzione** è davvero partito con essi (questo è quello che può restare indietro se non hai riavviato dopo averli abilitati), e il modulo kernel

stesso è operativo. Sotto: un grande pulsante enable/disable, e **ADVANCED · DEFAULT PROFILE MODE** — se i profili appena caricati partono in complain o enforce.

WHAT	AppArmor → STATUS → conferma che tutti e quattro i controlli siano verdi.
WHY	Se un controllo è rosso o giallo, nient'altro in questo pannello può funzionare correttamente — è sempre la prima cosa da guardare quando qualcosa con AppArmor sembra non andare.
RESULT	Certezza che il framework sia genuinamente attivo, non solo installato.
RISK	Nessuno — verifica in sola lettura.

7.2 Guida visiva — PROFILES

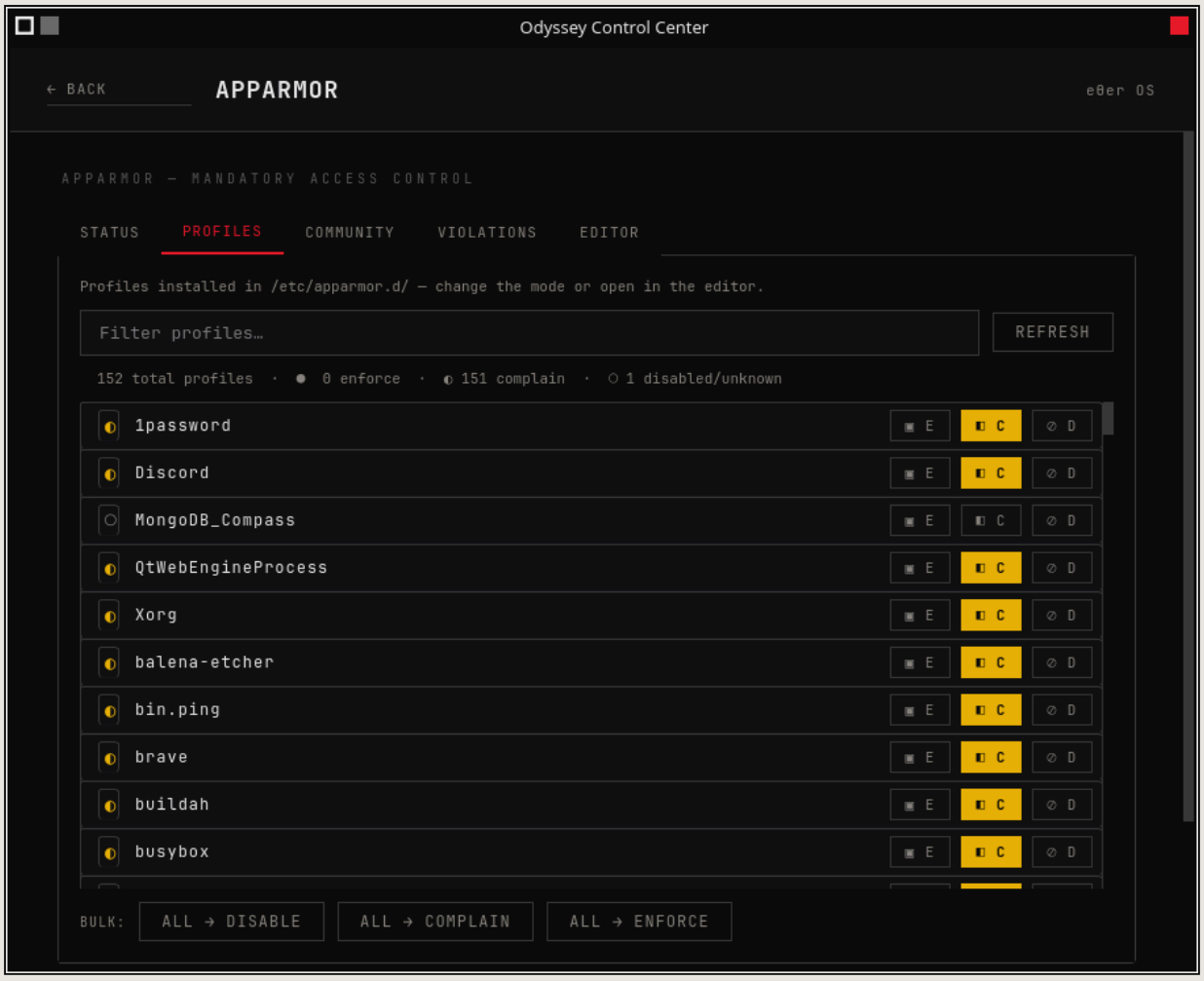

FIG · Tab PROFILES: 152 profili totali, 0 enforce, 151 complain, 1 disabled/unknown — una lista filtrabile con 1password, Discord, MongoDB_Compass e altri, ciascuno con pulsanti E / C / D, e in fondo ALL→DISABLE / ALL→COMPLAIN / ALL→ENFORCE.

Ogni profilo caricato, una riga ciascuno, con un campo filtro e un conteggio in cima (nello screenshot: 152 profili, tutti tranne uno in modalità complain — il default di Odyssey). Tre pulsanti per riga:

PULSANTE	EFFETTO
E	Passa questo profilo a enforce — le violazioni vengono bloccate, non solo registrate

PULSANTE	EFFETTO
C	Passa a complain — registra le violazioni, non blocca nulla
D	Disabilita il profilo del tutto — il programma gira non confinato

I tre pulsanti **BULK** in fondo applicano lo stesso passaggio a ogni profilo della lista in una volta — utile subito dopo aver installato un lotto da COMMUNITY, ma tratta ALL→ENFORCE con particolare cautela; vedi il blocco azione nella sezione successiva.

7.3 Guida visiva — COMMUNITY

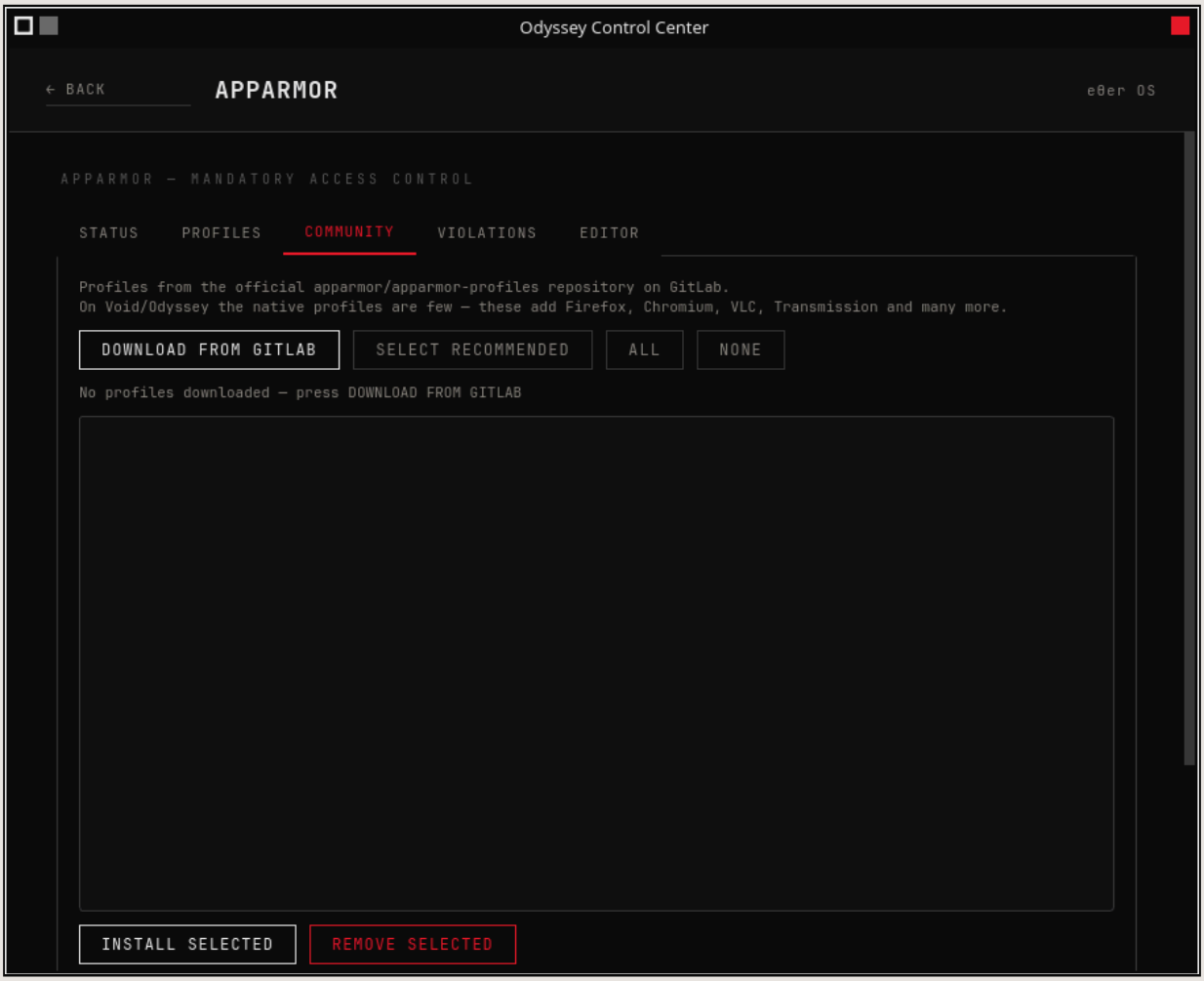

FIG · Tab COMMUNITY: “No profiles downloaded — press DOWNLOAD FROM GITLAB,” con i pulsanti DOWNLOAD FROM GITLAB, SELECT RECOMMENDED, ALL e NONE, e INSTALL SELECTED / REMOVE SELECTED in fondo.

AppArmor di serie confina soprattutto componenti di sistema — le applicazioni desktop di cui davvero ti preoccupi (browser, app di chat, media player) hanno bisogno di profili dalla community. Questa tab li recupera direttamente dal repository ufficiale `apparmor/apparmor-profiles` su GitLab.

1 **DOWNLOAD FROM GITLAB**

Recupera la lista attuale dei profili disponibili — nulla viene installato ancora, questo popola solo la lista.

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

Seleziona un insieme predefinito sensato per desktop — browser della famiglia Firefox, Chromium, VLC, Transmission e simili.

3 **INSTALL SELECTED**

Installa i profili selezionati — si caricano in modalità complain, come tutto il resto, pronti per il ciclo osserva-poi-applica sotto.

WHAT	AppArmor → COMMUNITY → DOWNLOAD FROM GITLAB → SELECT RECOMMENDED (o scegli singolarmente) → INSTALL SELECTED.
WHY	Le applicazioni più esposte a input non fidato — un browser che renderizza una pagina malevola, un media player che apre un file creato ad arte — sono esattamente quelle che un profilo protegge meglio.
RESULT	Nuovi profili compaiono in PROFILES, in modalità complain, registrando cosa avrebbero bloccato senza bloccare ancora nulla.
RISK	Nessuno all'installazione — niente viene applicato finché non lo decidi tu in PROFILES.

7.4 Guida visiva — VIOLATIONS

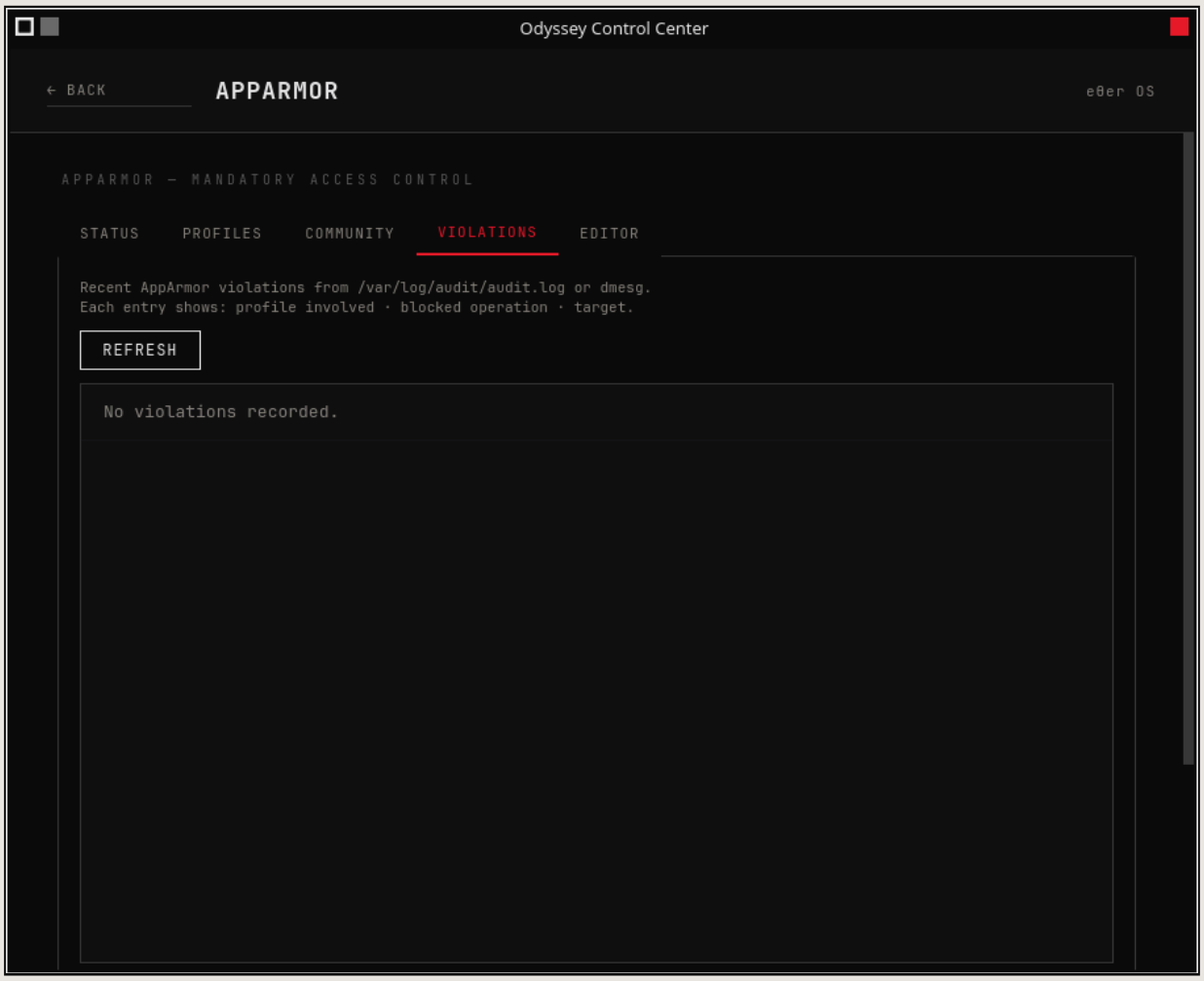

FIG · Tab VIOLATIONS: "No violations recorded," con un pulsante REFRESH e una nota che le voci provengono da /var/log/audit/audit.log o dmesg.

Questa tab legge il log di audit di AppArmor del sistema e mostra cosa un profilo **avrebbe** bloccato (in modalità complain) o **ha** bloccato (in modalità enforce) — ogni voce indica il profilo, l'operazione bloccata, e il file o la risorsa di destinazione.

> **UNA LISTA VUOTA È LA LETTURA ATTESA SU UN SISTEMA SANO**
>
> Lo screenshot mostra esattamente quello che vedrai tipicamente: nessuna violazione. Non è un bug né un segno che il pannello non funzioni — significa che ogni programma profilato è rimasto entro i limiti consentiti da quando il log è stato controllato l'ultima volta. Le violazioni compaiono qui nel momento in cui qualcosa di profilato prova a fare qualcosa che il suo profilo non permette; vai a usare il programma in questione, torna, e fai REFRESH.

Una volta che le voci compaiono, un doppio click su una di esse è la via rapida per entrare in EDITOR con quel profilo esatto già aperto — vedi sotto.

7.5 Guida visiva — EDITOR

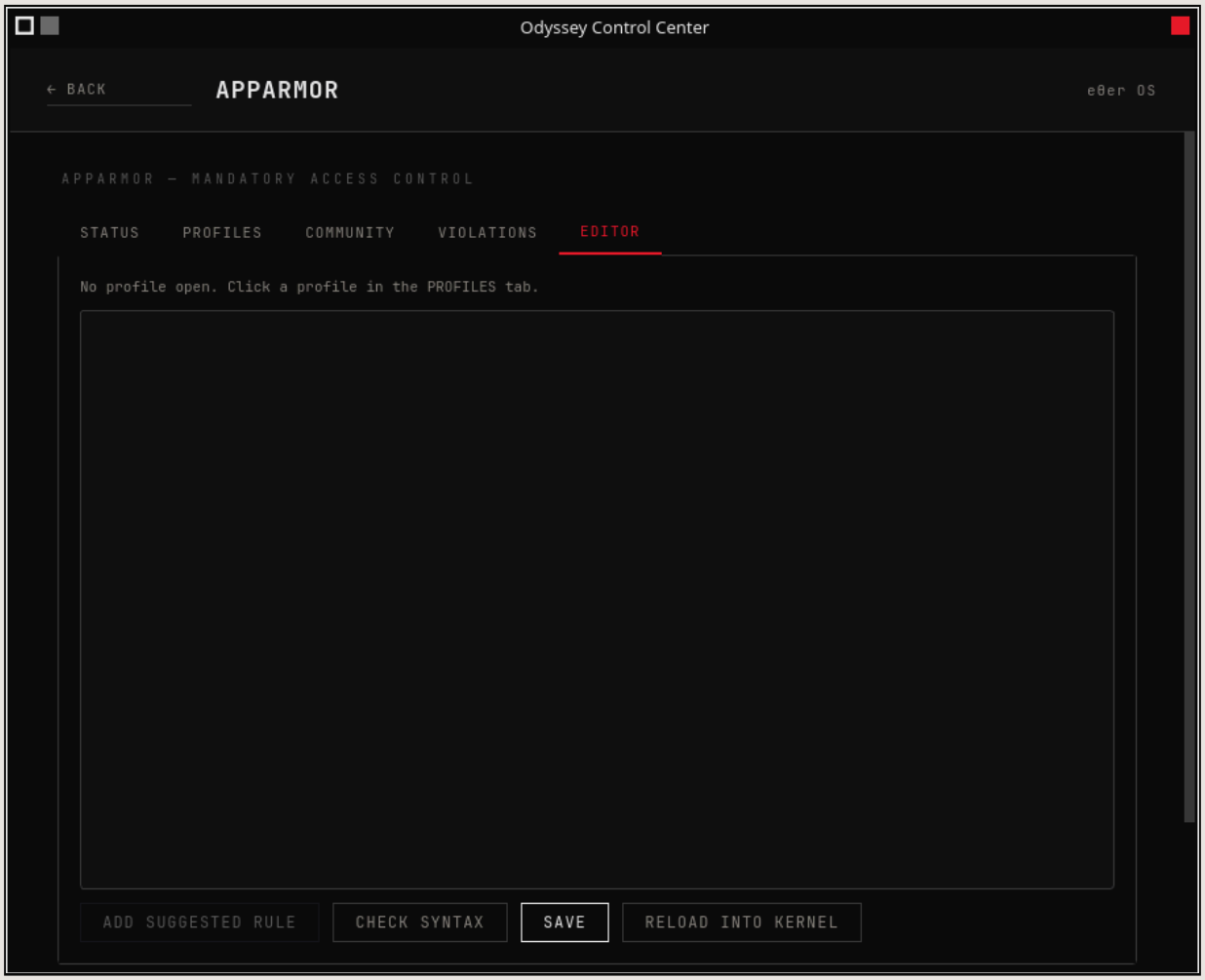

FIG · Tab EDITOR: "No profile open. Click a profile in the PROFILES tab," un'area di testo vuota, e quattro pulsanti — ADD SUGGESTED RULE, CHECK SYNTAX, SAVE, RELOAD INTO KERNEL — tutti tranne SAVE disattivati.

L'editor si apre deliberatamente **vuoto**, come mostrato — modifica un solo profilo alla volta, e scegli quale da PROFILES (clicca una riga) o da VIOLATIONS (doppio click su una voce). Questo previene l'errore facile di modificare il file sbagliato per sbaglio.

PULSANTE	COSA FA
ADD SUGGESTED RULE	Attivo solo se sei arrivato da una voce VIOLATIONS — inserisce la riga di regola esatta che avrebbe permesso l'azione registrata
CHECK SYNTAX	Convalida il profilo senza caricarlo — cattura un errore di battitura prima che conti
SAVE	Scrive le tue modifiche nel file del profilo su disco
RELOAD INTO KERNEL	Rende effettiva una modifica salvata immediatamente, senza aspettare il prossimo avvio o riavvio del servizio

WHAT	PROFILES → clicca il nome di un profilo → EDITOR lo apre → fai una modifica → CHECK SYNTAX → SAVE → RELOAD INTO KERNEL.
WHY	Questo è l'intero ciclo di confinamento in una riga: osserva in modalità complain (VIOLATIONS), correggi ciò che manca davvero (EDITOR), solo dopo passa quel profilo a enforce (PROFILES).
RESULT	Il profilo si comporta esattamente come modificato, live, senza riavvio.
RISK	Modificare un profilo in modo scorretto può far bloccare in seguito qualcosa di legittimo sotto enforce — CHECK SYNTAX cattura le regole malformate prima di SAVE, ma non gli errori logici. Se un profilo che hai modificato inizia a causare problemi sotto enforce, riportalo a complain da PROFILES e controlla di nuovo VIOLATIONS.

▸ ADVANCED

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

7.6 Advanced

EQUIVALENTI DA RIGA DI COMANDO

```
$ sudo aa-status
# gli stessi conteggi dell'intestazione della tab PROFILES, dal terminale
$ sudo aa-enforce /etc/apparmor.d/PROFILE_FILE
$ sudo aa-complain /etc/apparmor.d/PROFILE_FILE
$ sudo aa-disable /etc/apparmor.d/PROFILE_FILE
# equivalente a RELOAD INTO KERNEL dopo una modifica manuale
$ sudo apparmor_parser -r /etc/apparmor.d/PROFILE_FILE
```

LEGGERE IL LOG DI AUDIT GREZZO

```
$ sudo grep apparmor /var/log/audit/audit.log | tail -20
# oppure, se non è in esecuzione un audit daemon dedicato:
$ sudo dmesg | grep -i apparmor
```

Campi utili da conoscere leggendo una riga a mano: `apparmor="DENIED"` (enforce) o `apparmor="ALLOWED"` (complain, solo registrato), `operation=` (cosa è stato tentato), `profile=` (quale profilo ha deciso — corrisponde a ciò che apre EDITOR), `name=` (il percorso o la risorsa), `comm=` (il processo).

PERCHÉ ALL→ENFORCE IN BLOCCO È RISCHIOSO

Portare a enforce un profilo individualmente dopo che il suo log VIOLATIONS è rimasto silenzioso per un po' è il pattern sicuro — sai, profilo per profilo, che l'uso normale non lo fa scattare. ALL→ENFORCE salta questo periodo di osservazione per tutti i profili insieme, inclusi quelli che magari hai appena installato da COMMUNITY e mai davvero usato. Un profilo applicato prima di essere stato vissuto può bloccare una funzione legittima nel momento stesso in cui la usi per la prima volta. Preferisci il ciclo uno alla volta; riserva le azioni in blocco per ALL→COMPLAIN (sempre sicuro — allenta solo) o ALL→DISABLE quando fai troubleshooting.

FILE TOCCATI DA QUESTO PANNELLO

PERCORSO	RUOLO
`/etc/apparmor.d/`	Un file di profilo per ogni programma confinato — quello che elenca PROFILES e apre EDITOR
`/etc/default/apparmor`	Modalità di default al boot (`APPARMOR=complain`) e lo stato enable/disable
`/etc/default/grub`	I parametri kernel `apparmor=1 security=apparmor` che STATUS controlla

CHAPTER 8

Privacy

Due strumenti indipendenti, ciascuno col proprio flusso in tre step install → enable → running, e un collegamento tra i due una volta che entrambi sono attivi.

8.1 Guida visiva — SearXNG, passo per passo

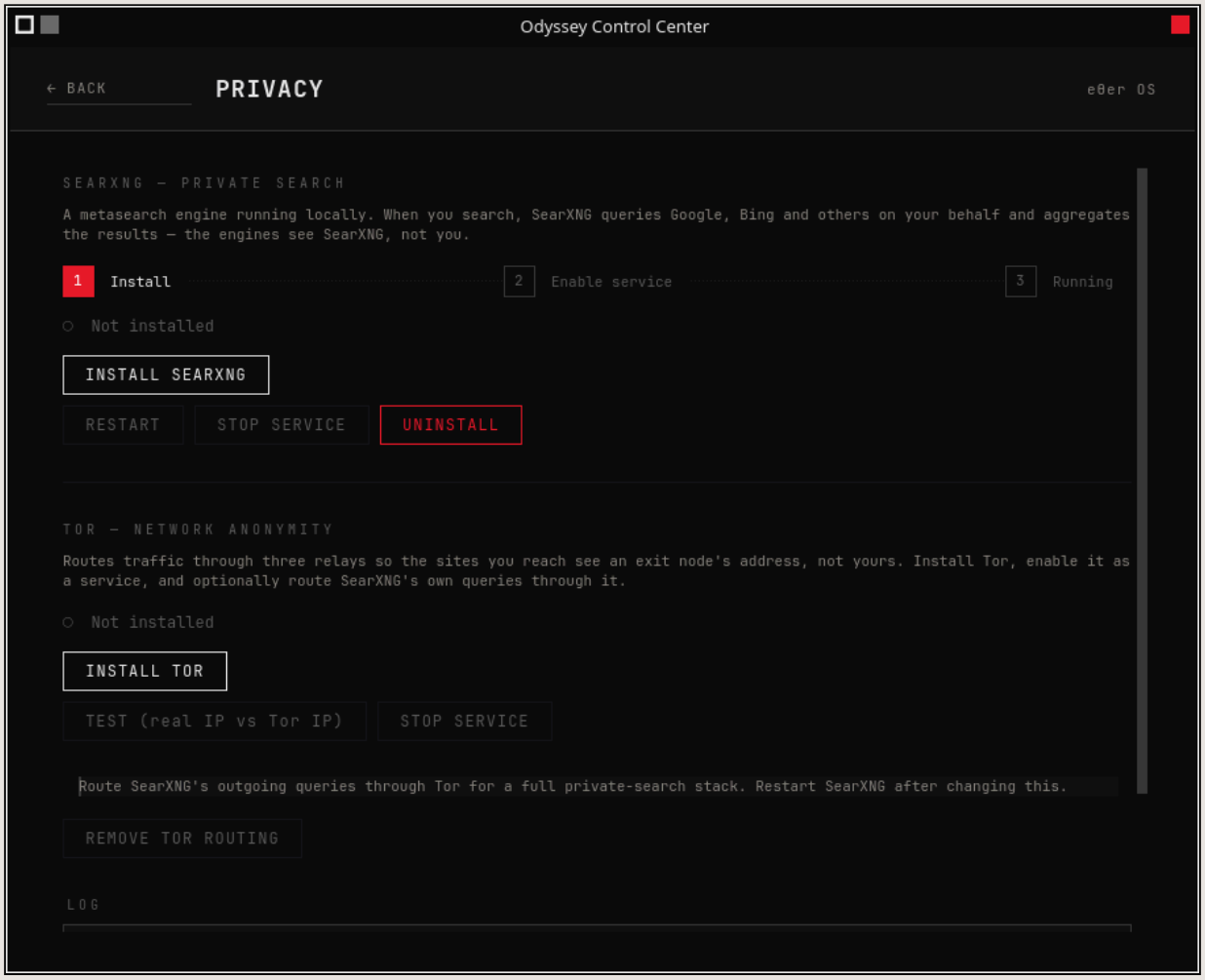

FIG · Il pannello a riposo: il tracker di SearXNG allo step 1 “Install,” stato “Not installed,” un solo pulsante INSTALL SEARXNG; anche il tracker di Tor mostra “Not installed” con INSTALL TOR sotto.

SearXNG è un motore di metaricerca che gira localmente sulla tua macchina — quando cerchi, è SearXNG a interrogare Google, Bing e gli altri per tuo conto, così quei motori vedono SearXNG fare la richiesta, mai te direttamente.

1 **INSTALL SEARXNG**

Cliccalo. Il log mostra l'installazione reale in corso — clonazione del sorgente, creazione di un utente di sistema, configurazione di un ambiente virtuale Python.

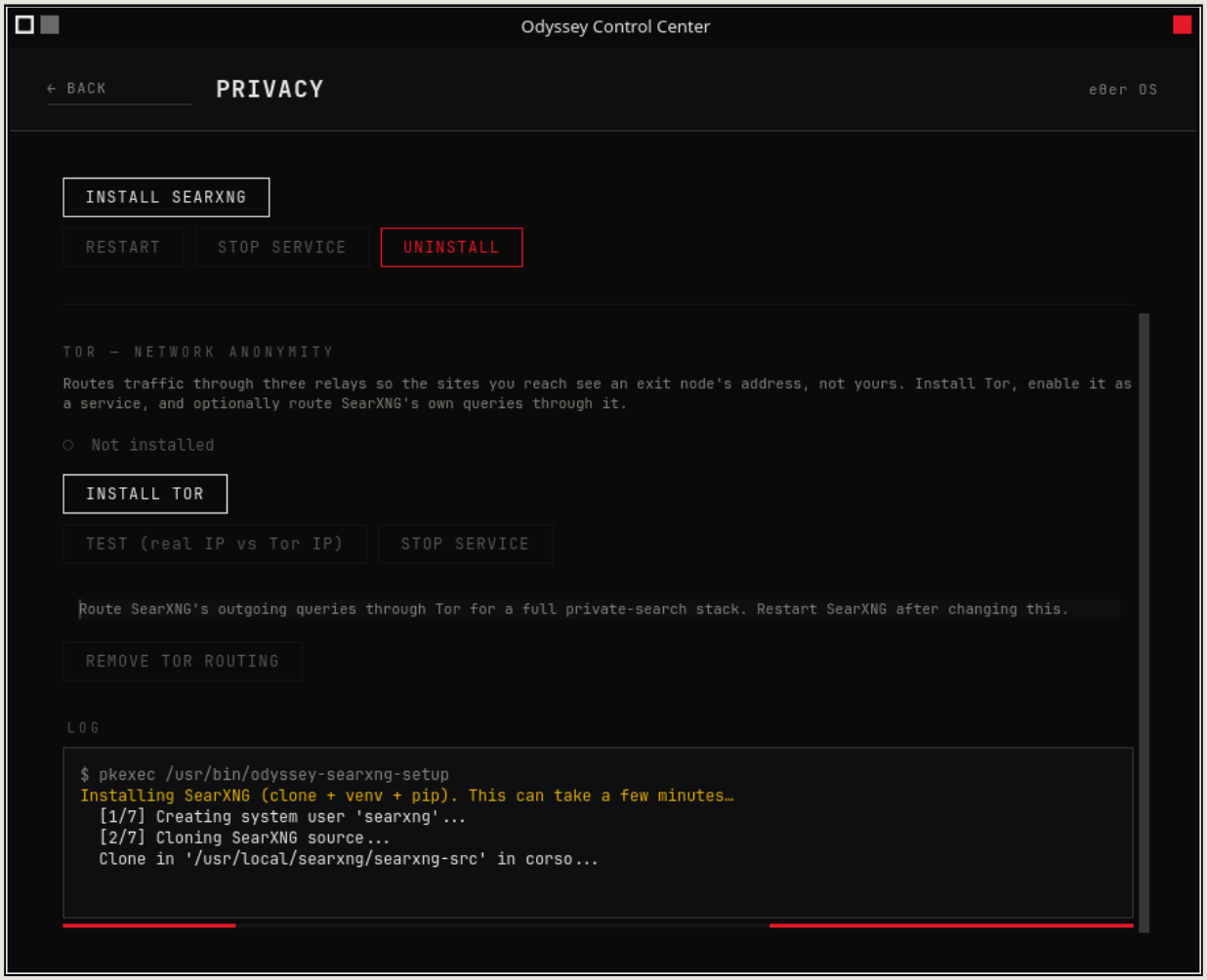

FIG · A metà installazione: il log che trasmette output reale — "Installing SearXNG (clone + venv + pip). This can take a few minutes…" con gli step 1/7 e 2/7 visibili.

2 ENABLE SERVICE

Una volta installato, il tracker passa allo step 2 e la riga di stato diventa gialla: "Installed — service not enabled." Un solo pulsante ora: ENABLE SERVICE.

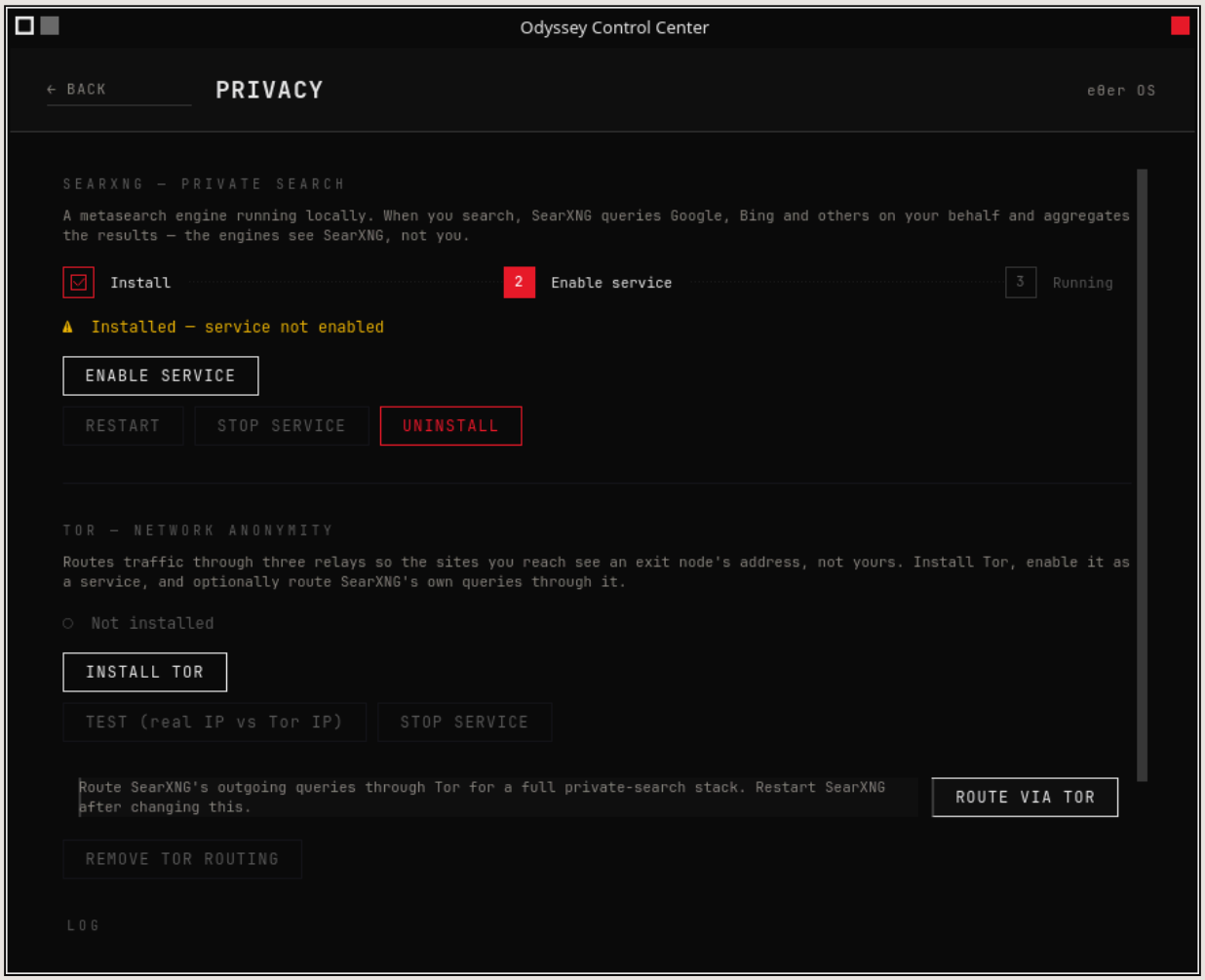

FIG · Step 2 attivo: “Installed — service not enabled,” ENABLE SERVICE come unico pulsante attivo, RESTART/STOP SERVICE/UNINSTALL ancora disattivati sotto.

3 Running

Dopo aver abilitato, il tracker raggiunge lo step 3 e l’intera riga di controllo si attiva.
Una volta attivo, **OPEN IN BROWSER** ti porta direttamente lì — SearXNG ascolta su `http://localhost:8888` per default.

WHAT	Privacy → INSTALL SEARXNG → ENABLE SERVICE → OPEN IN BROWSER.
WHY	Ogni motore di ricerca commerciale costruisce un profilo dalle tue query nel tempo; un’istanza di metaricerca locale significa che solo tu vedi mai la query non aggregata.
RESULT	Un motore di ricerca privato in esecuzione sulla tua macchina, raggiungibile nel browser, che interroga i grandi motori per tuo conto senza esporti a loro direttamente.
RISK	Basso — è un servizio web locale come un altro. RESTART se si comporta male, STOP SERVICE per metterlo in pausa senza rimuoverlo, UNINSTALL per rimuoverlo del tutto.

8.2 Guida visiva — Tor, passo per passo

Stessa forma in tre step, tracker indipendente, subito sotto la sezione di SearXNG.

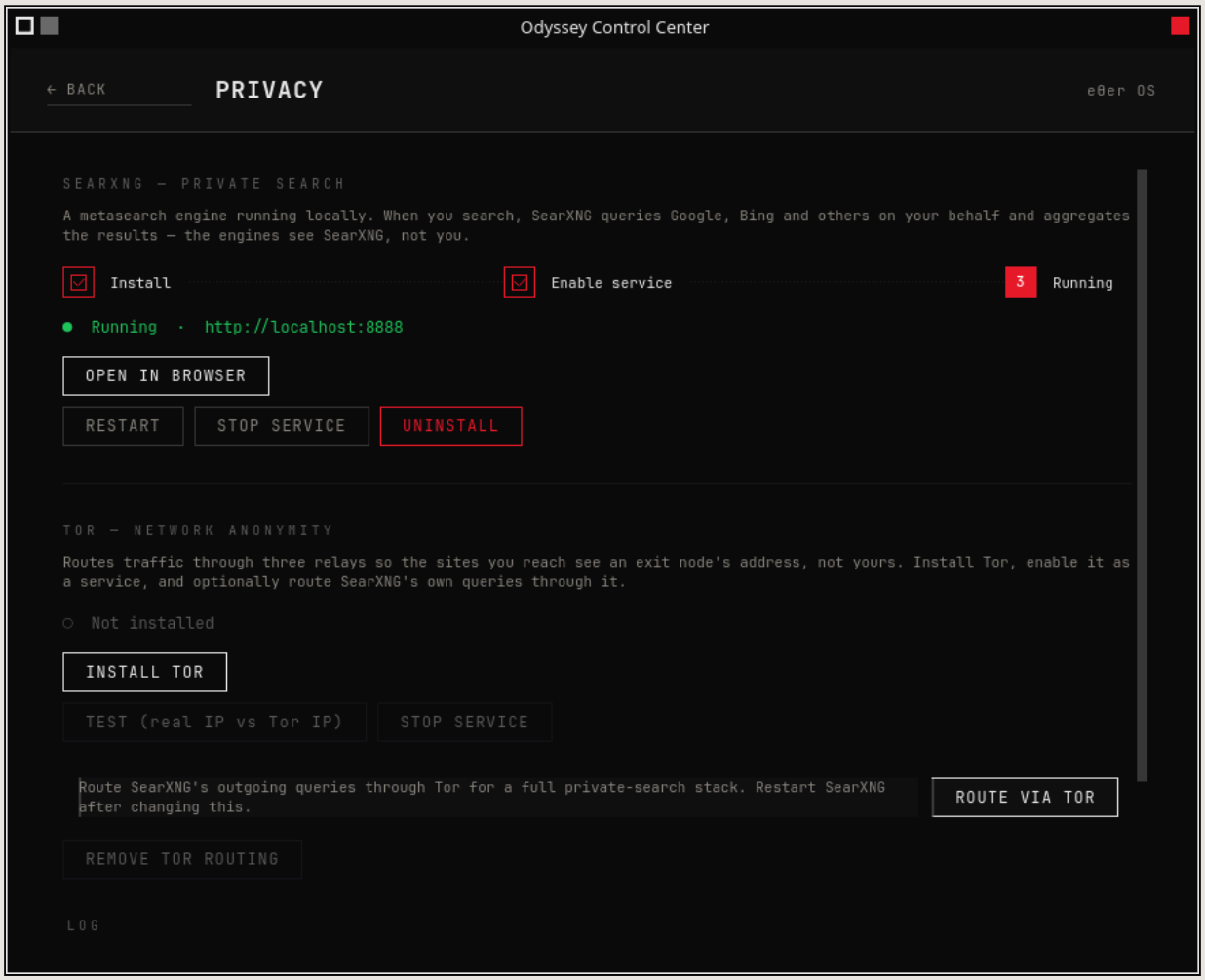

FIG · SearXNG pienamente attivo sopra; la sezione Tor sotto ancora a “Not installed” con INSTALL TOR come unico pulsante attivo.

Tor instrada il tuo traffico attraverso tre relay prima che raggiunga la destinazione, così il sito che visiti vede l’indirizzo del relay d’uscita, mai il tuo.

1 **INSTALL TOR**

Il log conferma l’installazione del pacchetto e mostra il servizio collegato in runit.

2 **ENABLE SERVICE**

Lo stato passa a “Installed — service not enabled” — stesso pattern di SearXNG.

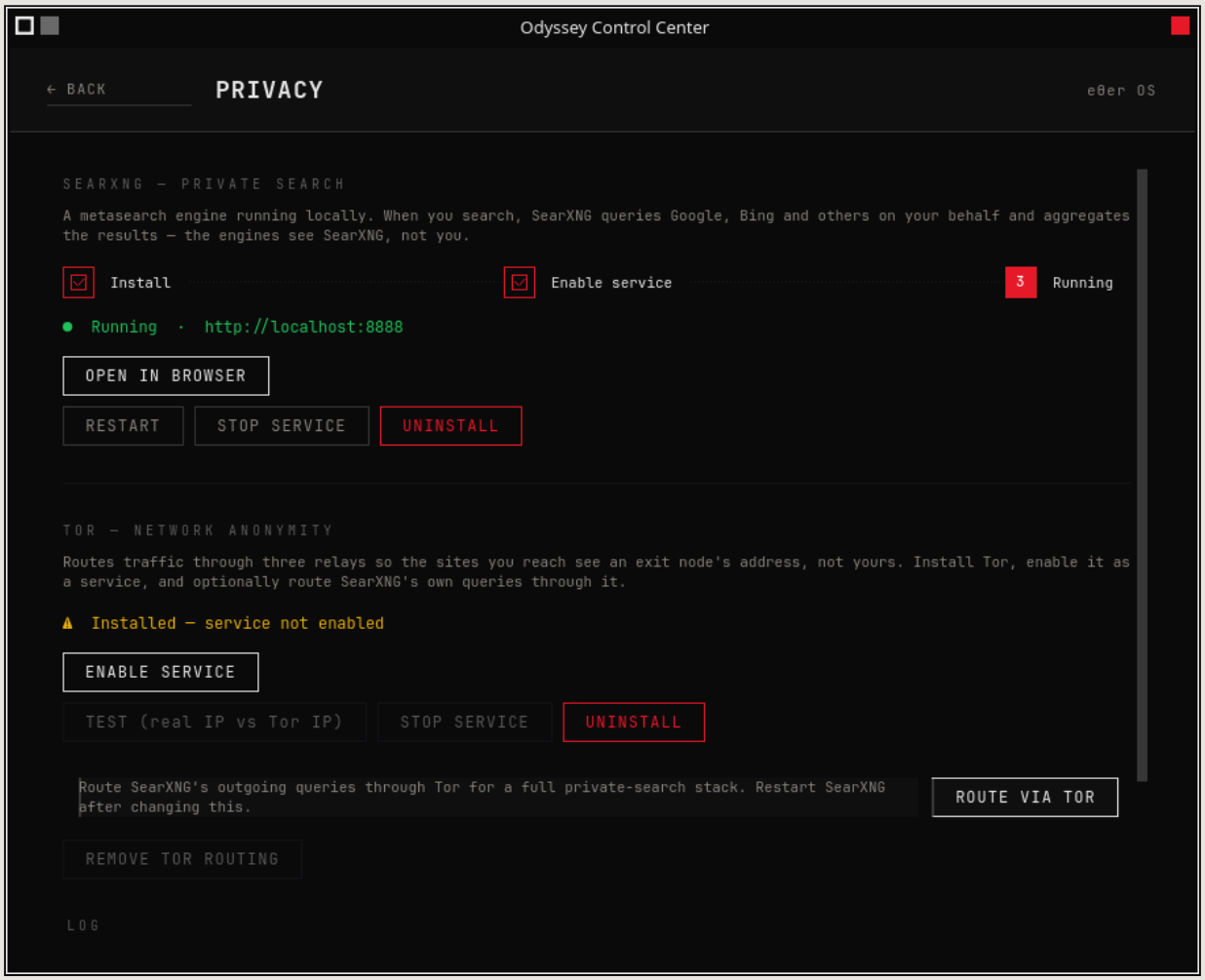

FIG · Tor allo step 2: “Installed — service not enabled,” ENABLE SERVICE attivo, TEST e STOP SERVICE ancora disattivati.

3 Running, e verificalo

Una volta abilitato, appare un pulsante dedicato **TEST (real IP vs Tor IP)** — usalo. Confronta il tuo indirizzo reale con quello che Tor sta presentando, fianco a fianco, così hai una prova reale che ha funzionato invece di una spia di stato a cui devi solo credere.

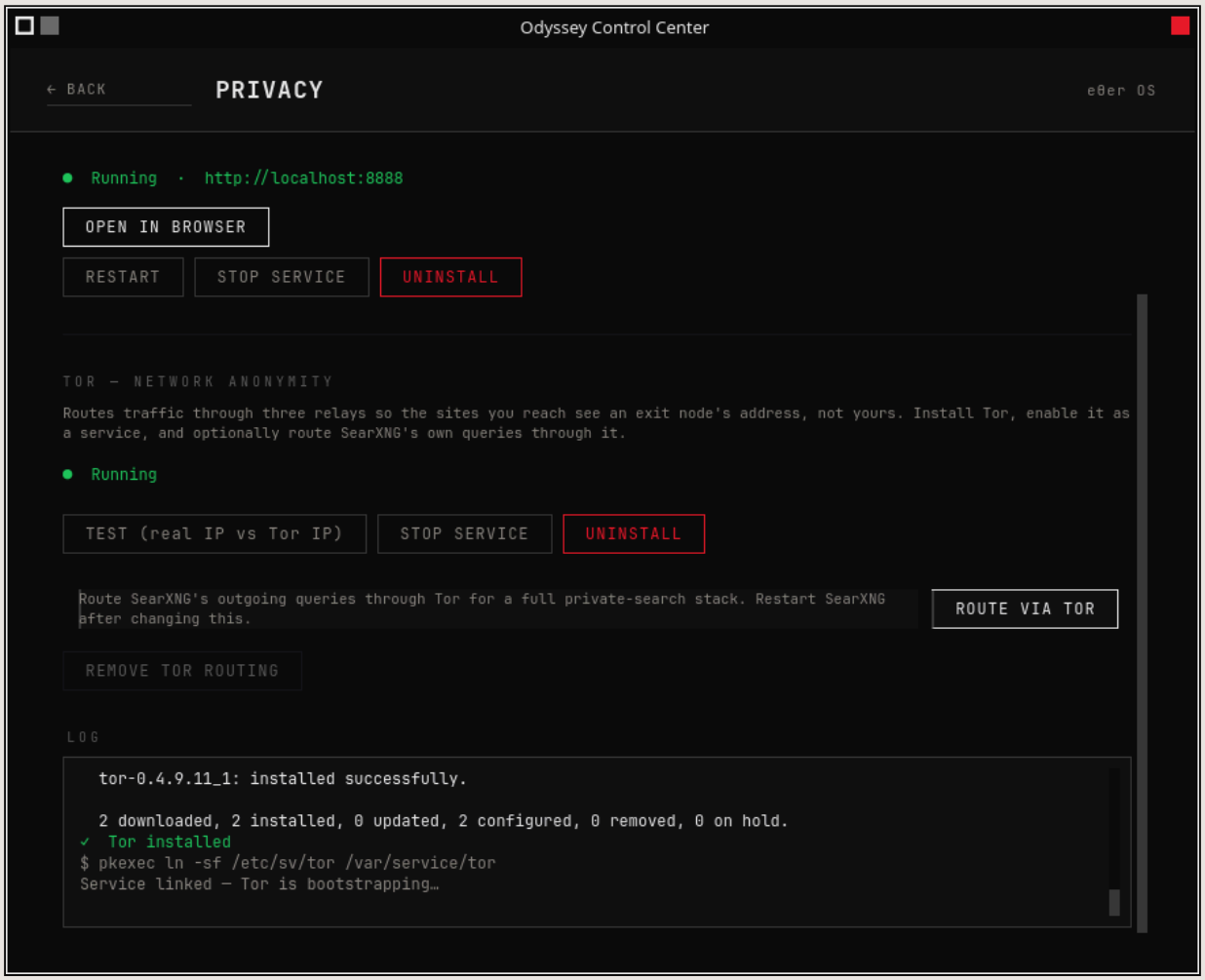

FIG · Entrambe le sezioni verdi e attive: SearXNG su localhost:8888 con OPEN IN BROWSER, Tor attivo con TEST / STOP SERVICE / UNINSTALL tutti disponibili.

WHAT	Privacy → INSTALL TOR → ENABLE SERVICE → TEST (real IP vs Tor IP) per confermare.
WHY	Instradare la navigazione sensibile attraverso Tor nasconde il tuo indirizzo ai siti che visiti — TEST ti dà una prova, non solo un'affermazione.
RESULT	Un servizio Tor in esecuzione attraverso cui altre applicazioni possono instradarsi; TEST mostra due indirizzi genuinamente diversi se sta funzionando.
RISK	Il traffico attraverso Tor è decisamente più lento — è il prezzo per l'anonimato. STOP SERVICE lo mette in pausa; UNINSTALL lo rimuove.

8.3 Guida visiva — combinarli

Una volta che entrambi sono attivi, appare un avviso in linea che offre di instradare anche le query **proprie** di SearXNG attraverso Tor — non solo le tue richieste a SearXNG, ma le richieste di SearXNG a Google e Bing per tuo conto.

WHAT	Privacy → l'avviso "Route SearXNG's outgoing queries through Tor" → ROUTE VIA TOR → RESTART sulla sezione SearXNG perché abbia effetto.
WHY	Senza questo, SearXNG stesso continua a interrogare i grandi motori di ricerca direttamente — privato dal tuo lato, visibile dal loro. Instradare il traffico proprio di SearXNG attraverso Tor chiude questo divario per uno stack di ricerca completamente privato.
RESULT	Le query in uscita di SearXNG, non solo le tue query verso di esso, passano attraverso Tor.
RISK	I risultati si caricano decisamente più lentamente, stesso motivo di qualunque traffico instradato via Tor. REMOVE TOR ROUTING lo annulla, di nuovo seguito da un riavvio.

▸ ADVANCED

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

8.4 Advanced

COSA ESEGUE DAVVERO INSTALL SEARXNG

```
$ pkexec /usr/bin/odyssey-searxng-setup
# crea un utente di sistema dedicato 'searxng',
# clona il sorgente in /usr/local/searxng/searxng-src,
# costruisce un venv Python e installa le dipendenze con pip
```

SERVIZI E PORTE, A MANO

```
# il servizio runit di SearXNG
$ sudo sv status searxng
$ sudo sv restart searxng
# il servizio runit di Tor — lo step di abilitazione del pannello è letteralmente questo
$ sudo ln -sf /etc/sv/tor /var/service/tor
$ sudo sv status tor
# verifica che Tor stia davvero instradando
$ curl -socks5 localhost:9050 https://check.torproject.org/api/ip
```

INSTRADARE SEARXNG ATTRAVERSO TOR A MANO

Il proxy SOCKS di Tor ascolta su `127.0.0.1:9050` per default. Instradare SearXNG attraverso di esso significa puntare il suo client HTTP in uscita verso quel proxy — l'impostazione vive nel file `settings.yml` di SearXNG stesso, nella sezione di configurazione delle richieste in uscita, piuttosto che in una variabile di proxy a livello di sistema. Il pulsante ROUTE VIA TOR del pannello modifica direttamente quel file e riavvia il servizio; farlo a mano significa individuare e modificare lo stesso blocco `outgoing:` in `/usr/local/searxng/searxng-src/searx/settings.yml`.

PERCHÉ SEARXNG HA BISOGNO DI UN RIAVVIO DOPO LE MODIFICHE DI INSTRADAMENTO

SearXNG legge la propria configurazione proxy una sola volta, all'avvio — non è un'impostazione ricaricabile a caldo. Questo vale sia che tu la cambi attraverso il pannello sia a mano, motivo per cui entrambi i percorsi finiscono con lo stesso passaggio RESTART.

FILE TOCCATI DA QUESTO PANNELLO

PERCORSO	RUOLO
`/usr/local/searxng/searxng-src/`	L'installazione stessa di SearXNG — sorgente, venv, config
`/etc/sv/searxng`, `/etc/sv/tor`	Definizioni dei servizi runit
`/var/service/`	Dove abilitare un servizio lo collega nell'albero di supervisione in esecuzione

CHAPTER 9

Containers

Gestione dei container end to end: configurazione del motore, una lista dei container in esecuzione, un catalogo curato da cui partire, comandi grezzi per qualunque cosa non ci sia dentro, e uso del disco — tutto in un unico pannello scorrevole.

9.1 Guida visiva — il motore

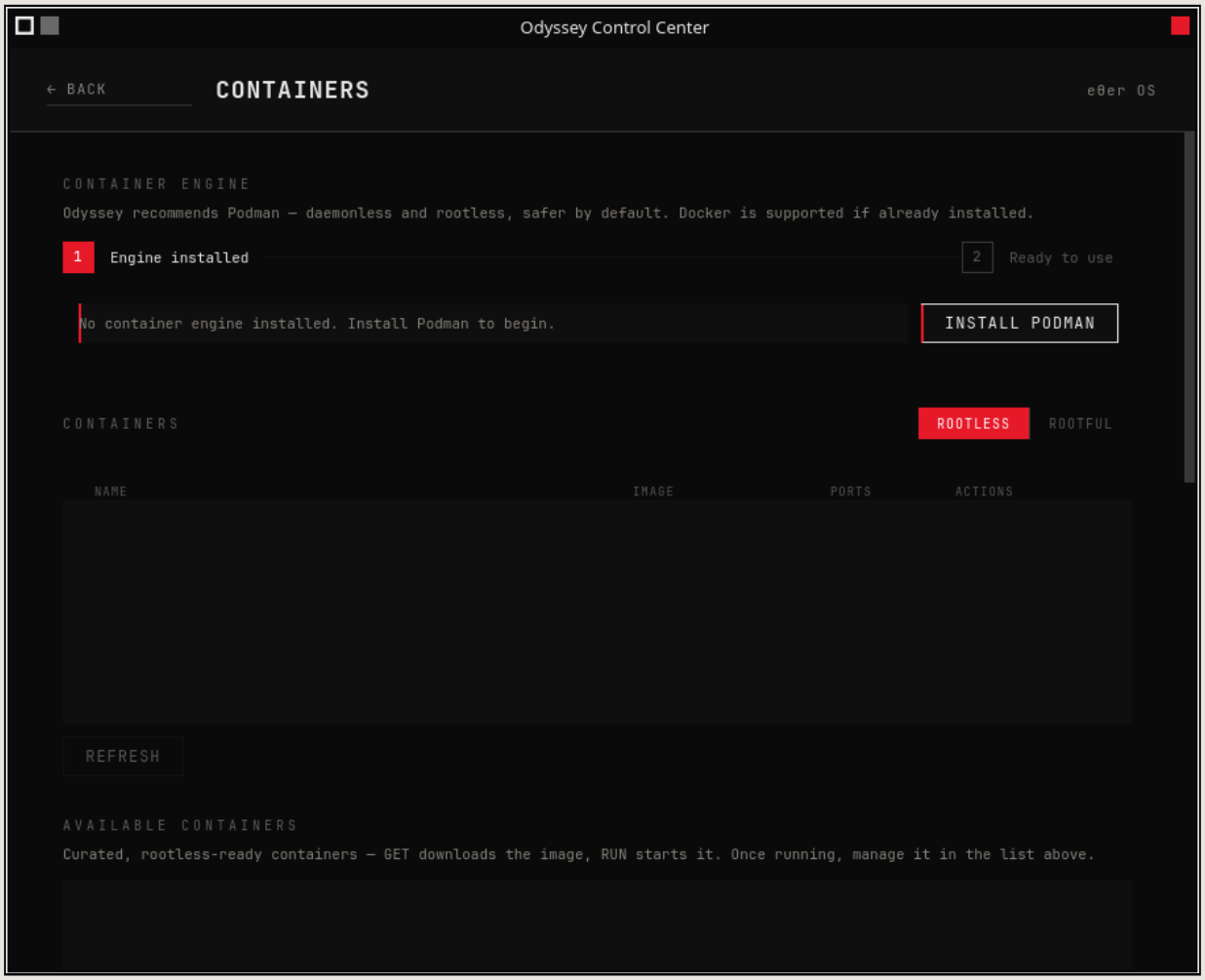

FIG · Step 1 di 2: “No container engine installed. Install Podman to begin,” con INSTALL PODMAN come unica azione attiva. Le sezioni Containers list e Available containers sotto sono visibili ma in stato vuoto.

Odyssey consiglia **Podman** — senza demone e rootless, il che significa che i container girano senza un servizio in background permanente e, per default, senza privilegi di root. **Docker** è supportato anche se è già quello che usi.

1 **INSTALL PODMAN**

Il log mostra l’installazione reale del pacchetto.

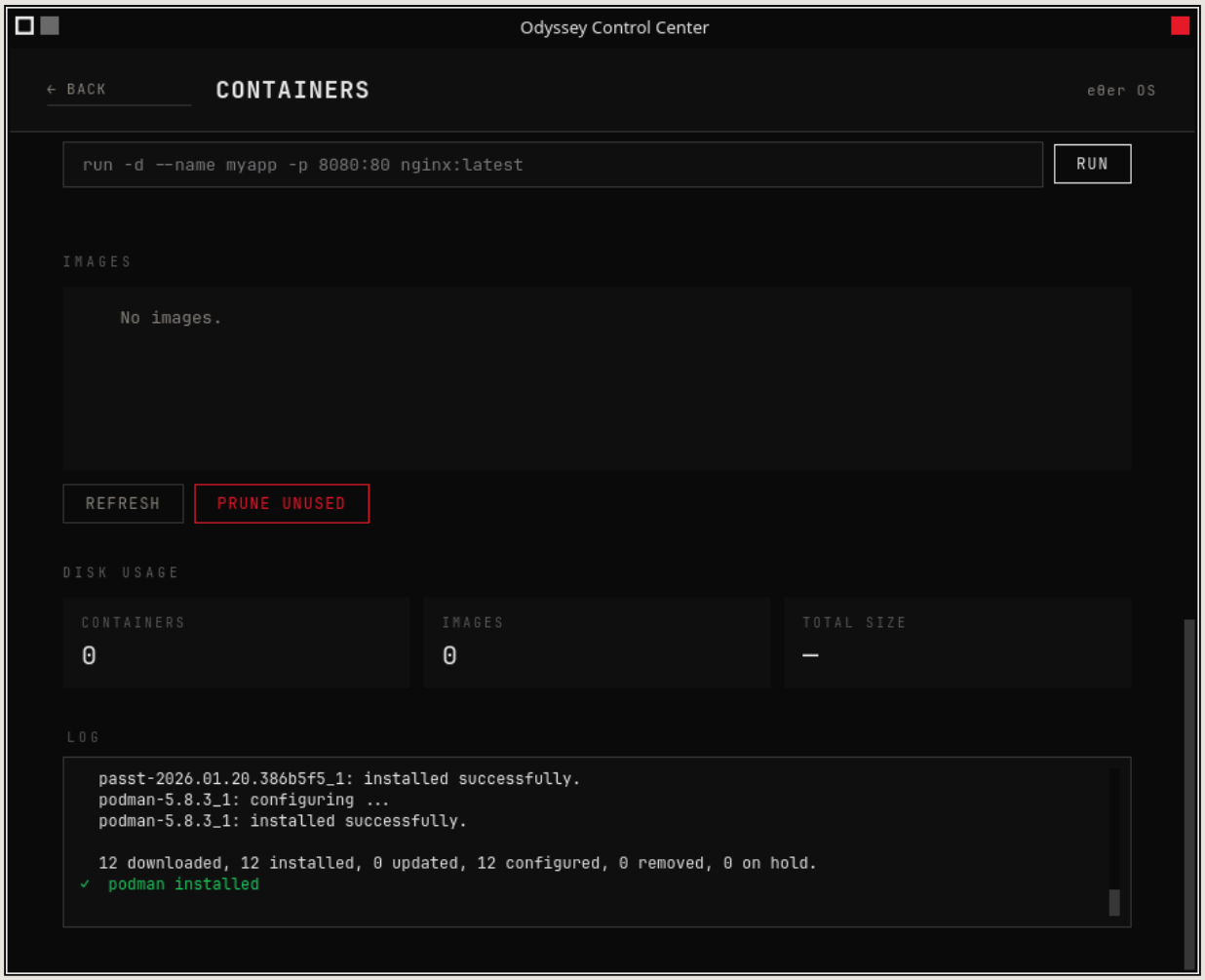

FIG · Dopo l’installazione: “Engine installed” spuntato in rosso, step 2 “Ready to use,” la riga informativa che legge “podman version 5.8.3 · Podman installed · Docker not installed,” e un pulsante INSTALL DOCKER TOO per chi vuole anche Docker in parallelo.

2 Ready to use

Il motore è attivo. **INSTALL DOCKER TOO** resta disponibile se vuoi entrambi i motori fianco a fianco — coesistono senza conflitti.

9.2 Guida visiva — rootless vs. rootful, e lo stato vuoto

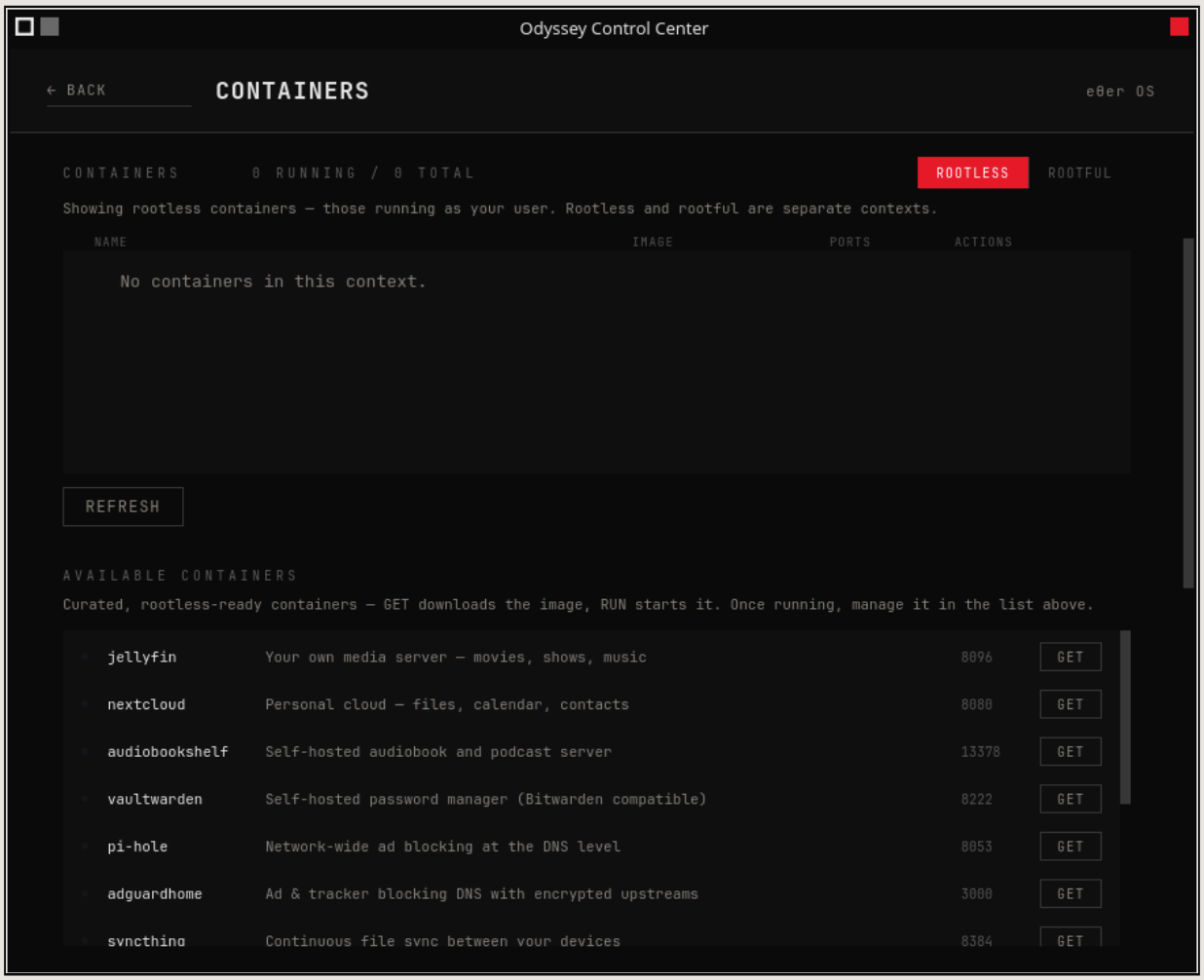

FIG · ROOTLESS selezionato (evidenziato in rosso) accanto a ROOTFUL, “0 RUNNING / 0 TOTAL,” e “No containers in this context.” sotto le intestazioni di colonna NAME / IMAGE / PORTS / ACTIONS.

L’interruttore **ROOTLESS / ROOTFUL** in cima cambia quale contesto stai visualizzando e gestendo — i due sono namespace di container completamente separati, non un filtro su un’unica lista combinata. Odyssey mostra ROOTLESS per default, e quel default è deliberato:

> **QUALE DEI DUE DOVREI USARE**
>
> Rootless è la scelta più sicura ed è quella verso cui il pannello ti indirizza: un container che sfugge al confinamento in modalità rootless non può comunque fare nulla che il tuo stesso account utente non potesse già fare. Rootful esiste per i casi specifici che ne hanno bisogno — certe porte di rete con numero basso, alcuni accessi hardware — e dovrebbe essere l’eccezione, non l’abitudine.

9.3 Guida visiva — il catalogo curato, dall’inizio alla fine

Scorri fino a **Available containers** — una lista già pronta così il pannello non è mai una scatola vuota in attesa che tu conosca già i nomi delle immagini a memoria. Ogni riga mostra il container, una descrizione in una riga, e la porta che usa.

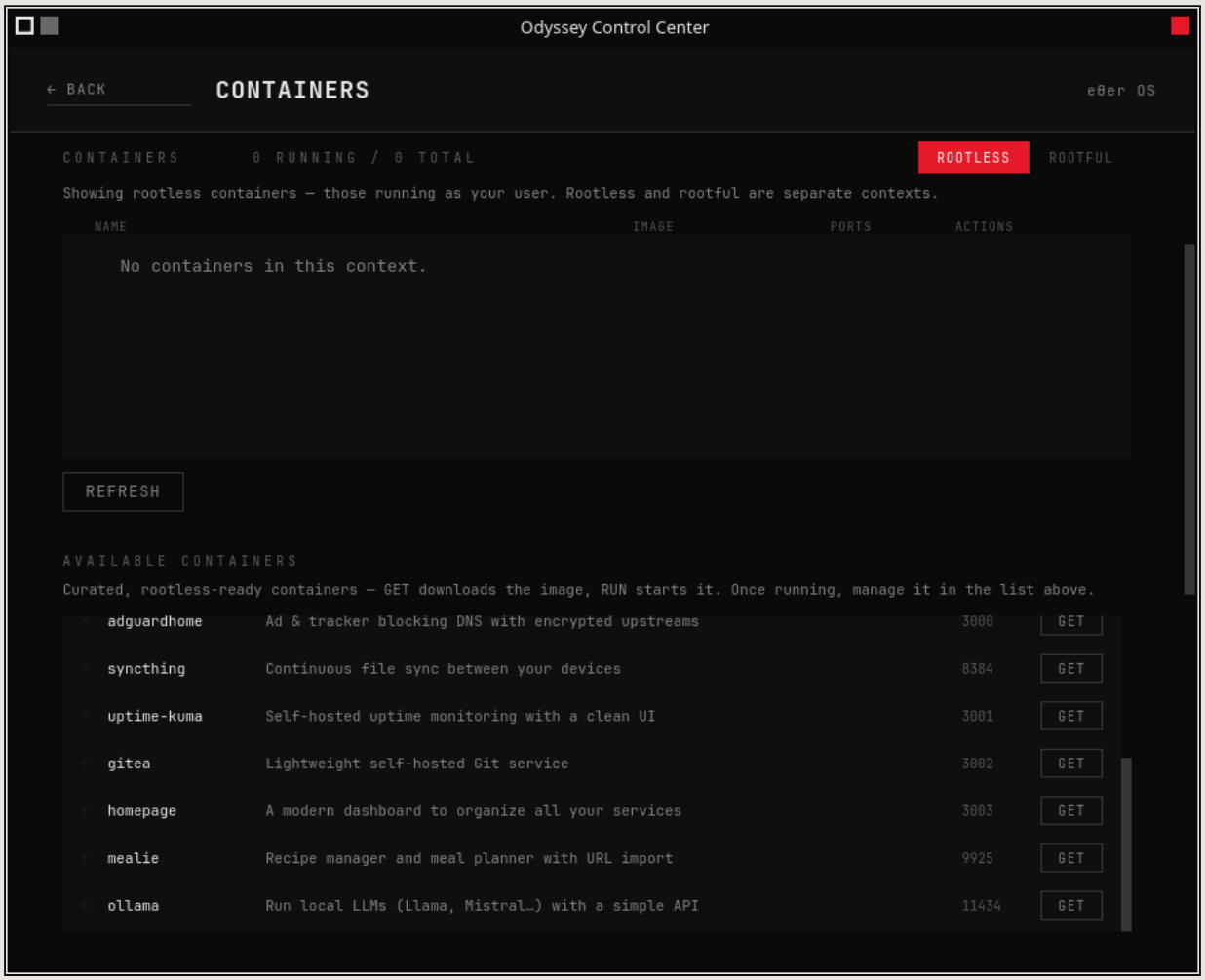

FIG · La lista del catalogo: adguardhome (porta 3000), syncthing (8384), uptime-kuma (3001), gitea (3002), homepage (3003) — con un pulsante GET su ogni riga.

Quel numero di porta è quello su cui raggiungerai il servizio una volta in esecuzione — `homepage` con `3003`, per esempio, significa `http://localhost:3003` una volta attivo.

1 GET

Clicca GET su `homepage`. Questo scarica solo l'immagine del container — niente è ancora in esecuzione.

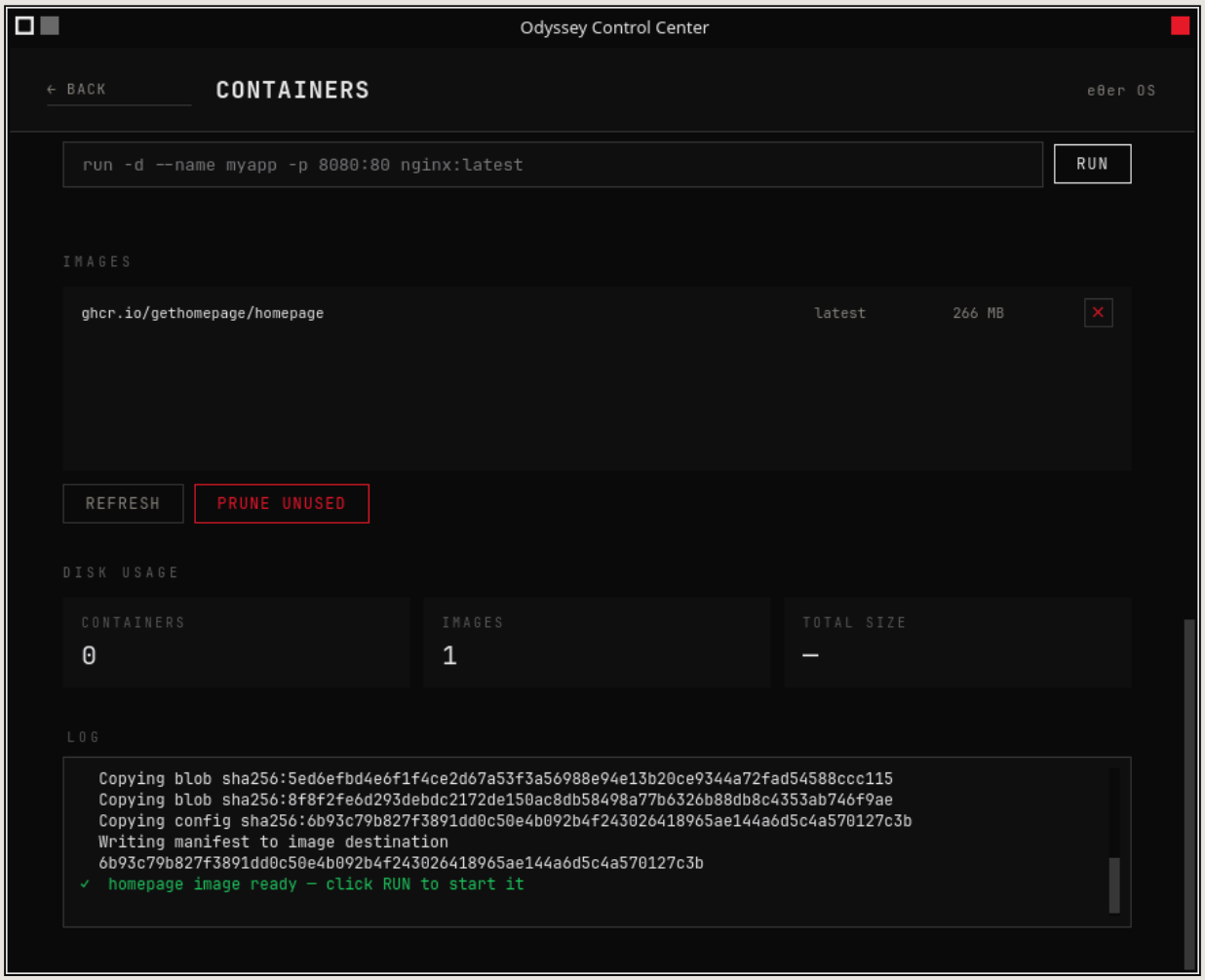

FIG · Il log che trasmette uno scaricamento reale dell'immagine: "Copying blob sha256:5ed6…", "Writing manifest to image destination," terminando in verde — "homepage image ready — click RUN to start it."

2 RUN

La riga del pulsante GET ora mostra RUN al suo posto — cliccalo per avviare davvero il container.

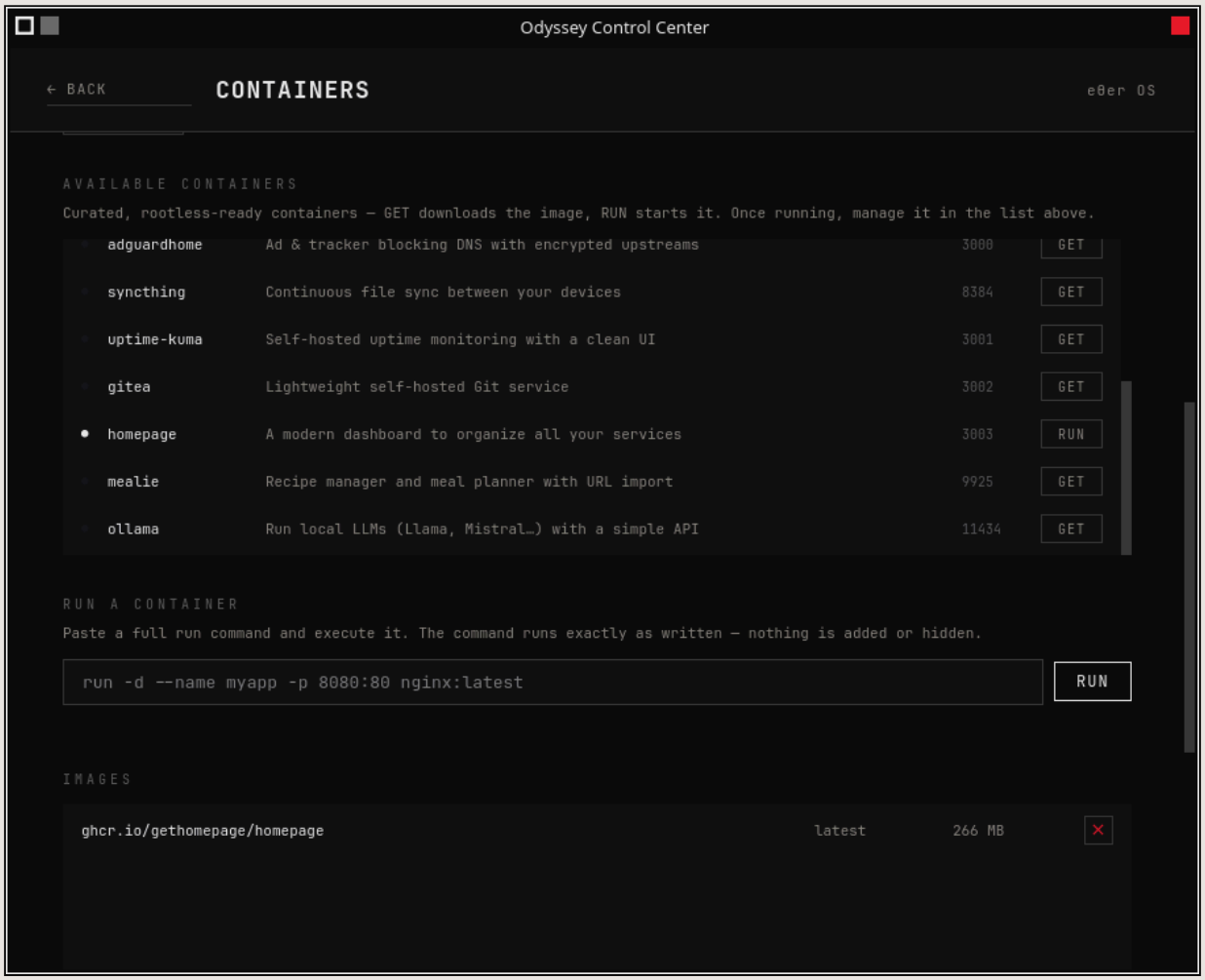

FIG · homepage ora elencato nella sezione CONTAINERS sopra, pallino verde, percorso immagine ghcr.io/gethomepage/homepage, porte "0.0.0.0:3003→…", con la propria riga di icone azione; ancora visibile sotto in Available Containers con un nuovo pulsante RUN per avviare un'altra istanza.

Una volta in esecuzione, `homepage` compare in due posti contemporaneamente: come voce attiva nella lista **Containers** in cima (con la sua porta e le icone azione), e ancora nel catalogo sotto nel caso tu voglia avviare una seconda istanza più avanti.

WHAT	Containers → Available containers → GET su qualcosa nel catalogo → RUN una volta scaricato.
WHY	La divisione in due step (scarica, poi avvia) significa che puoi recuperare un'immagine in anticipo senza impegnarti ad avviarla subito — utile su una connessione lenta o quando stai solo esplorando cosa è disponibile.
RESULT	Un container in esecuzione, visibile nella lista principale con la sua mappatura di porta, gestibile da lì in avanti.
RISK	Basso — i servizi self-hosted del catalogo curato sono scelti per essere rootless-friendly e a basso rischio per default. L'esposizione è la stessa di eseguire qualunque servizio web: qualunque porta ascolti è raggiungibile secondo le tue impostazioni Firewall.

9.4 Guida visiva — la lista dei container e le sue azioni

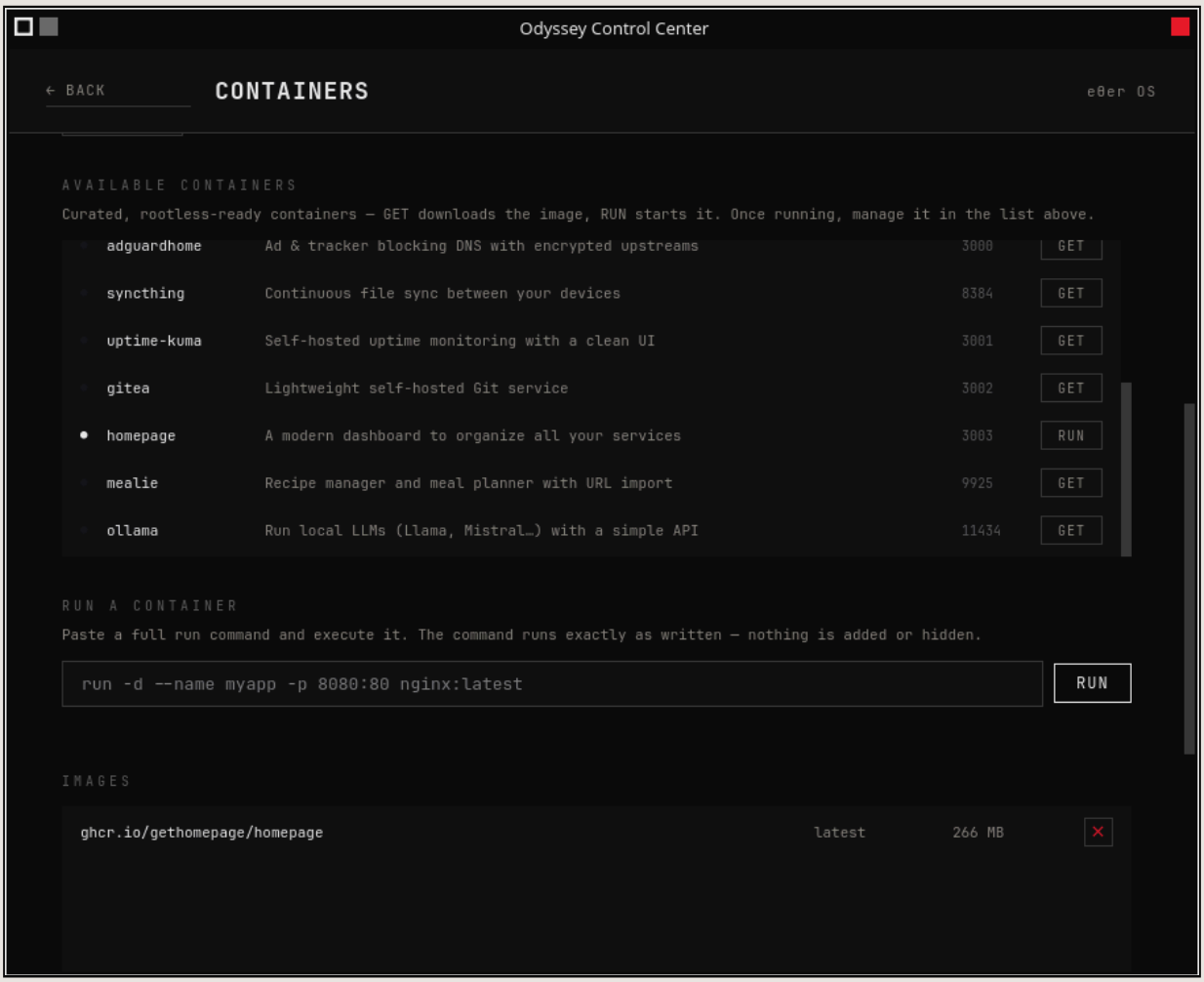

FIG · La riga di homepage nella lista Containers, con le cinque icone azione visibili a destra — quelle spiegate sotto.

Ogni riga di un container in esecuzione termina in un insieme di icone — passa il mouse su ognuna per vederne l'etichetta, ma ecco cosa fa ciascuna:

ICONA	AZIONE
apri (freccia)	Apre il servizio direttamente nel tuo browser — la via rapida una volta che qualcosa è in esecuzione
stop / avvia (quadrato / triangolo)	Ferma il container senza rimuoverlo, oppure lo riavvia
riavvia (freccia circolare)	Lo ferma e riavvia in un'unica azione — la soluzione per "è in esecuzione ma non risponde"
log (icona a righe)	Apre l'output di log del container stesso — il primo posto da guardare se qualcosa non funziona
rimuovi (×)	Rimuove del tutto il container — l'immagine stessa resta scaricata, elencata separatamente sotto Images

9.5 Guida visiva — eseguire un container, e aprirlo

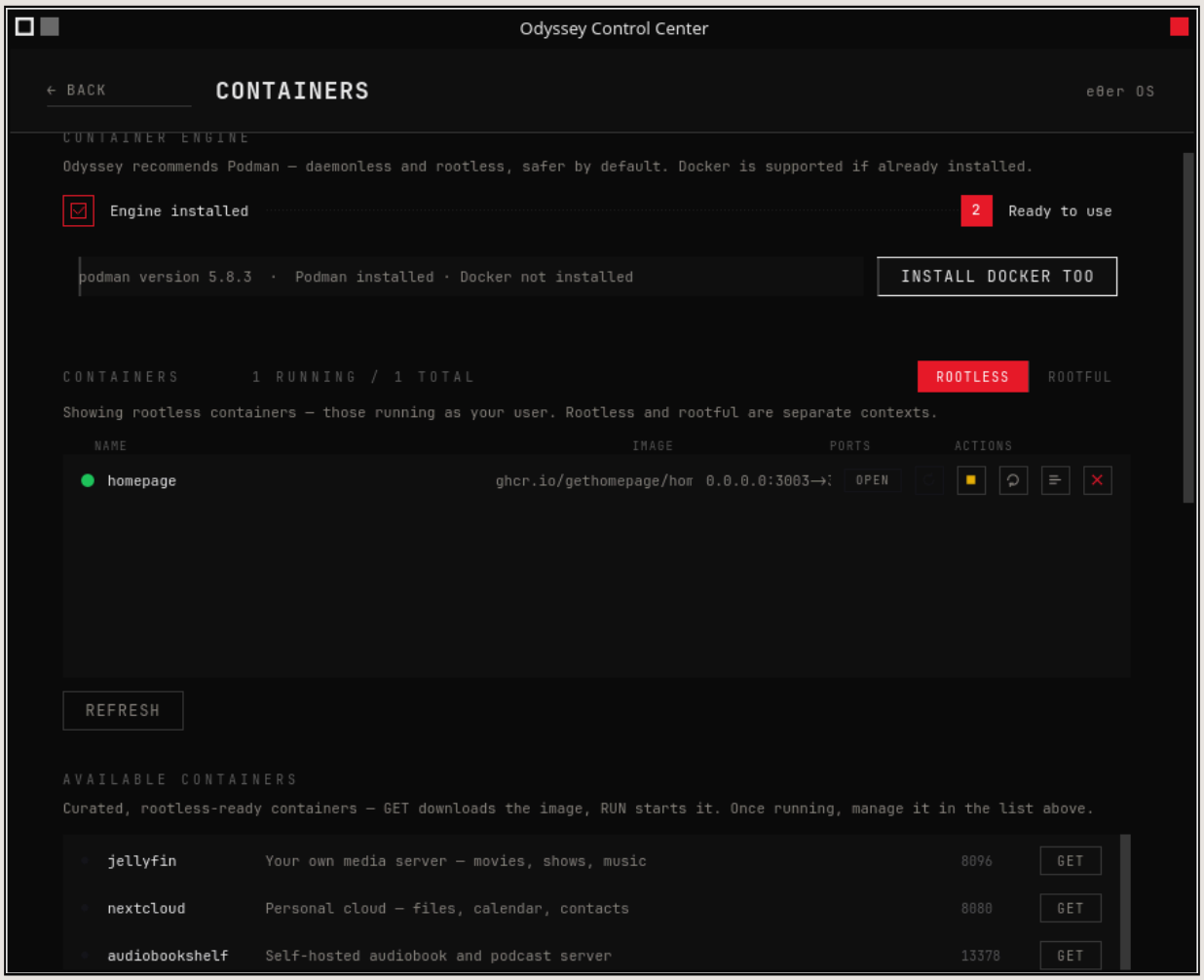

FIG · Il campo “Run a container” con un esempio di comando come placeholder, il pulsante RUN, e la sezione Images sotto che elenca ghcr.io/gethomepage/homepage · latest · 266 MB con un pulsante di rimozione.

Per qualunque cosa non nel catalogo, **Run a container** prende un comando `podman run` (o `docker run`) completo e lo esegue esattamente come scritto — niente viene aggiunto, niente nascosto.

WHAT	Containers → “Run a container” → incolla un comando run completo (es. `run -d --name myapp -p 8080:80 nginx:latest`) → RUN.
WHY	Il catalogo curato copre i casi comuni; questa è la via di fuga per un’immagine specifica, flag specifici, porte specifiche che il catalogo non offre.
RESULT	Il container si avvia esattamente come specificato dal comando, e si unisce alla lista principale Containers una volta in esecuzione.
RISK	Qualunque cosa faccia il comando stesso — questo è un passaggio fedele, non una sandbox. Trattalo con la stessa cura con cui digiteresti lo stesso comando in un terminale.

Sotto, **Images** elenca ogni immagine scaricata indipendentemente dal fatto che un container sia attualmente in esecuzione da essa, con **REFRESH** e **PRUNE UNUSED** per recuperare spazio dalle immagini che nessuno referenzia più.

APRIRE UN CONTAINER IN ESECUZIONE

Cliccare OPEN sulla riga di un container in esecuzione — la stessa azione apri dalla riga di icone sopra — lancia il tuo browser di default direttamente al suo indirizzo e porta, es. `http://localhost:3003` per `homepage`. Nessun bisogno di ricordare tu stesso il numero di porta una volta che è in esecuzione.

9.6 Uso del disco

In fondo al pannello, tre totali in tempo reale — container, immagini, dimensione totale — così sai sempre cosa ti costano i container sul disco senza aprire un terminale.

▸ ADVANCED

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

9.7 Advanced

EQUIVALENTI DA RIGA DI COMANDO

```
# quello che il pannello chiama "GET"
$ podman pull ghcr.io/gethomepage/homepage
# quello che il pannello chiama "RUN" su una voce del catalogo
$ podman run -d -name homepage -p 3003:3000 ghcr.io/gethomepage/homepage
# elenca i container in esecuzione — contesto rootless
$ podman ps
# lo stesso, rootful
$ sudo podman ps
# log, stop, riavvia, rimuovi — la riga di icone, a mano
$ podman logs -f homepage
$ podman stop homepage
$ podman restart homepage
$ podman rm homepage
# immagini e uso del disco
$ podman images
$ podman system df
$ podman image prune
```

ROOTLESS VS. ROOTFUL, SOTTO IL COFANO

Podman rootless esegue i container come il tuo stesso utente, dentro un namespace utente — il "root" del container viene mappato su un intervallo di UID senza privilegi sull'host, così anche un processo container completamente compromesso resta comunque vincolato ai permessi del tuo stesso account. Podman rootful (`sudo podman ...`) funziona nel modo tradizionale, con il root del container mappato sul vero root dell'host. L'interruttore del pannello sta davvero solo scegliendo tra eseguire `podman` semplice o `sudo podman` — directory di stato realmente separate, liste di container separate, niente condiviso tra i due contesti per default.

```
# dove ogni contesto tiene il proprio stato
$ podman info -format '{{.Store.GraphRoot}}'
# rootless:  /.local/share/containers/storage
# rootful: /var/lib/containers/storage
```

PERCHÉ ALCUNE VOCI DEL CATALOGO HANNO BISOGNO DI PORTE SPECIFICHE APERTE

La mappatura di porta di un container (`3003:3000` nell'esempio homepage) rende il servizio raggiungibile **solo da questa macchina** per default — Podman lo lega a `localhost` a meno che non venga detto diversamente. Raggiungerlo da un altro dispositivo sulla tua rete richiede in aggiunta che quella porta sia aperta nel pannello Firewall (Capitolo 4) — i due sistemi sono livelli indipendenti, e un container in esecuzione con una porta mappata non è automaticamente raggiungibile da internet o dalla LAN solo perché Podman ci sta ascoltando.

CHAPTER 10

Gaming Mode

Un interruttore principale in cima, una riga di stato in tempo reale, e quattro tab: **PACKAGES**, **GAMEMODE**, **MANGOHUD**, **PROTON**.

10.1 Guida visiva — l'interruttore principale

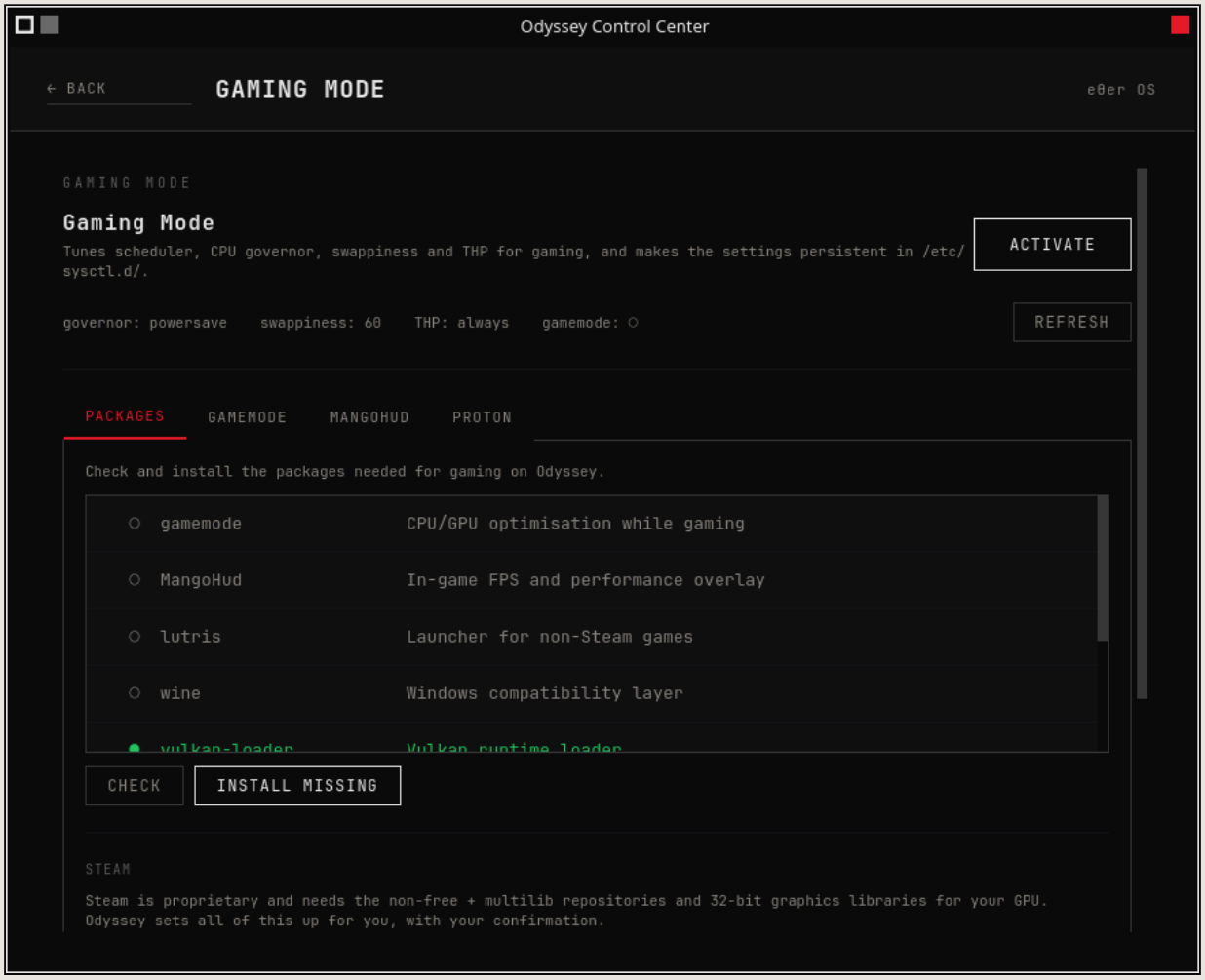

FIG · La sezione in cima al pannello: "Gaming Mode — Tunes scheduler, CPU governor, swappiness and THP for gaming, and makes the settings persistent in /etc/sysctl.d/," un pulsante ACTIVATE, un pulsante REFRESH, e la riga di stato live: governor powersave, swappiness 60, THP always, gamemode off.

La riga di stato mostra gli stessi valori live che i pannelli Kernel e Memory mostrano — governor, swappiness, stato THP, e se il servizio `gamemode` è in esecuzione — così puoi confermare a colpo d'occhio se l'attivazione ha davvero funzionato, senza visitare separatamente nessuno dei due pannelli.

WHAT	Gaming Mode → ACTIVATE.
WHY	Visitare manualmente Kernel e Memory per impostare a mano un governor, una swappiness e un valore THP adatti al gaming è esattamente il tipo di lavoro a più passaggi che una modalità dedicata dovrebbe eliminare.
RESULT	Scheduler, governor, swappiness e THP passano tutti a valori adatti al gaming, applicati immediatamente e resi persistenti — la riga di stato si aggiorna per riflettere il cambiamento.
RISK	Basso, e reversibile — gli stessi valori che imposteresti altrimenti a mano in Kernel/Memory, solo raggruppati in un click.

10.2 Guida visiva — Packages e Steam

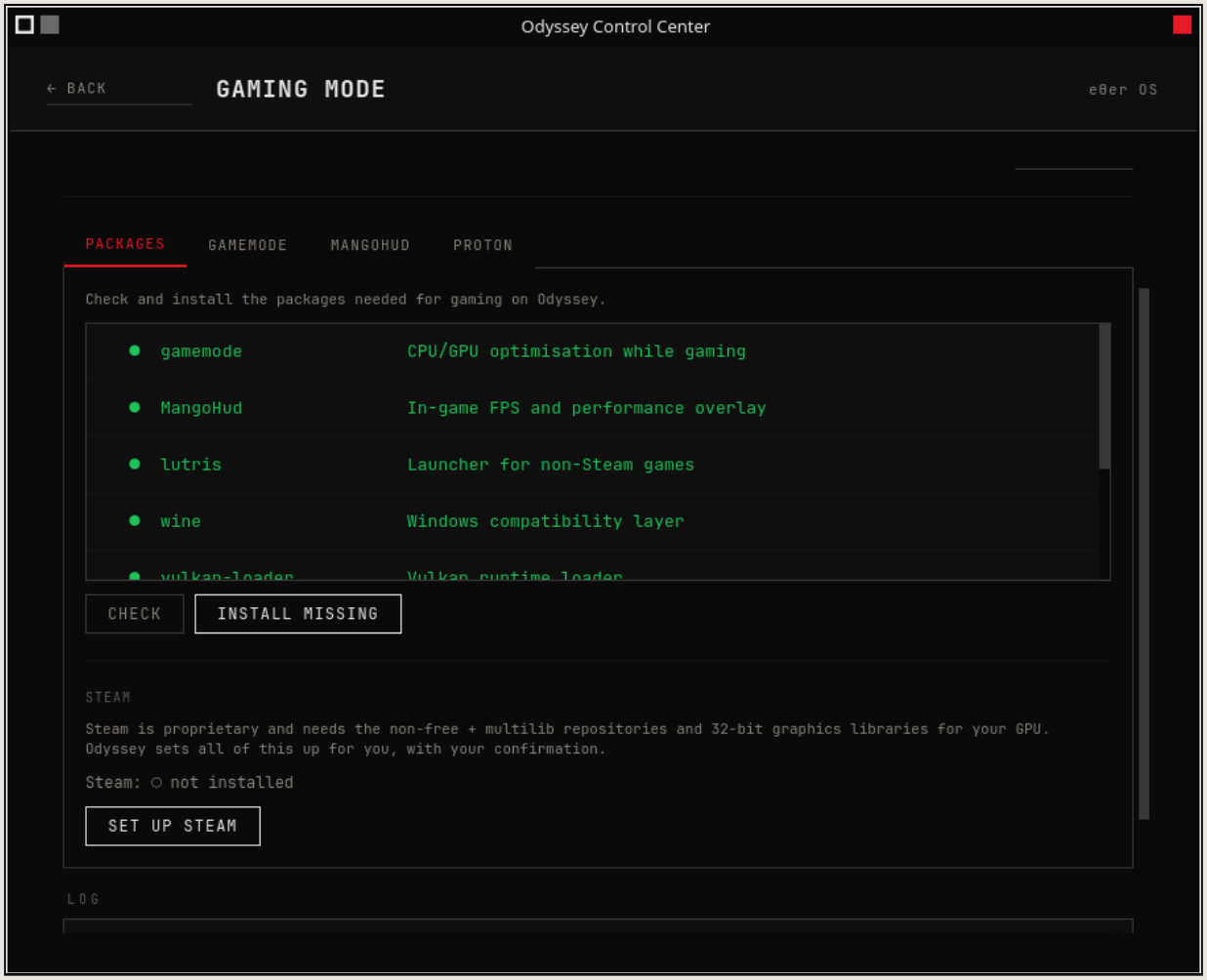

FIG · La tab PACKAGES: gamemode, MangoHud, lutris, wine, vulkan-loader tutti con pallini verdi pieni (installati); CHECK e INSTALL MISSING sopra; la sezione Steam sotto con "Steam: ● installed" e un pulsante REINSTALL STEAM.

Una checklist dei pacchetti generalmente necessari per il gaming — CHECK vede cosa manca, INSTALL MISSING colma le lacune in un'unica azione. Una volta che tutto mostra un pallino verde pieno, come nello screenshot, sei completamente configurato.

Steam ha un proprio flusso dedicato sotto la checklist, perché ha bisogno di più di una semplice installazione di pacchetto.

1 **SET UP STEAM**

Cliccalo — non installa ancora nulla, apre prima un dialogo di conferma informativo.

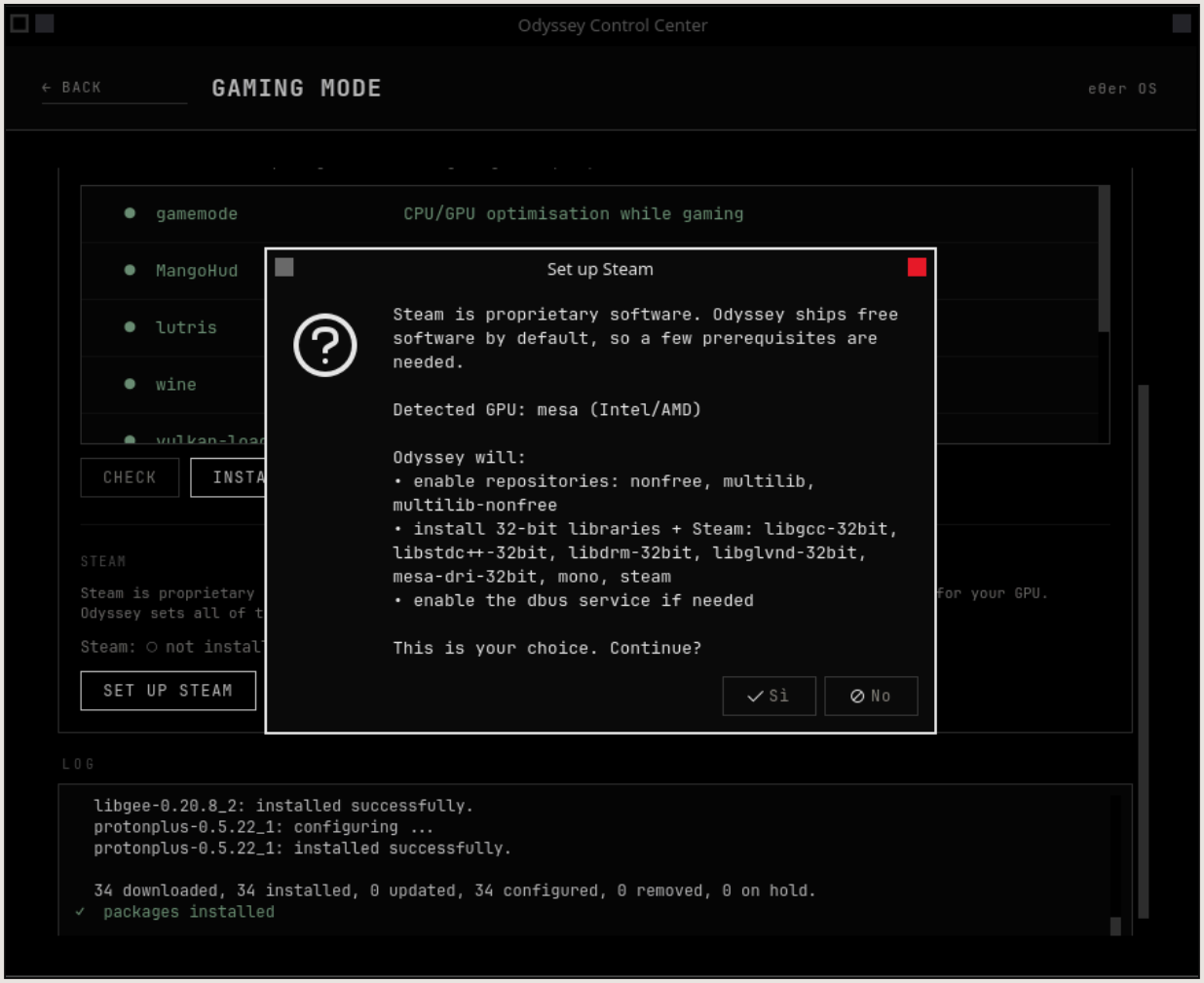

FIG · Il dialogo di conferma: “Steam is proprietary software. Odyssey ships free software by default, so a few prerequisites are needed. Detected GPU: mesa (Intel/AMD).” Un elenco puntato di cosa verrà esattamente abilitato e installato, che termina “This is your choice. Continue?” con i pulsanti Sì / No.

Il dialogo ti dice esattamente cosa sta per succedere prima che tu acconsenta: quali repository vengono abilitati (`nonfree`, `multilib`, `multilib-nonfree`), quali librerie a 32-bit vengono installate in base alla GPU **rilevata**, e che il servizio dbus verrà abilitato se non lo è già. Niente è nascosto dietro la conferma.

2 **Sì**

La conferma. Il log trasmette l’installazione reale — scaricamento pacchetti, configurazione, completamento.

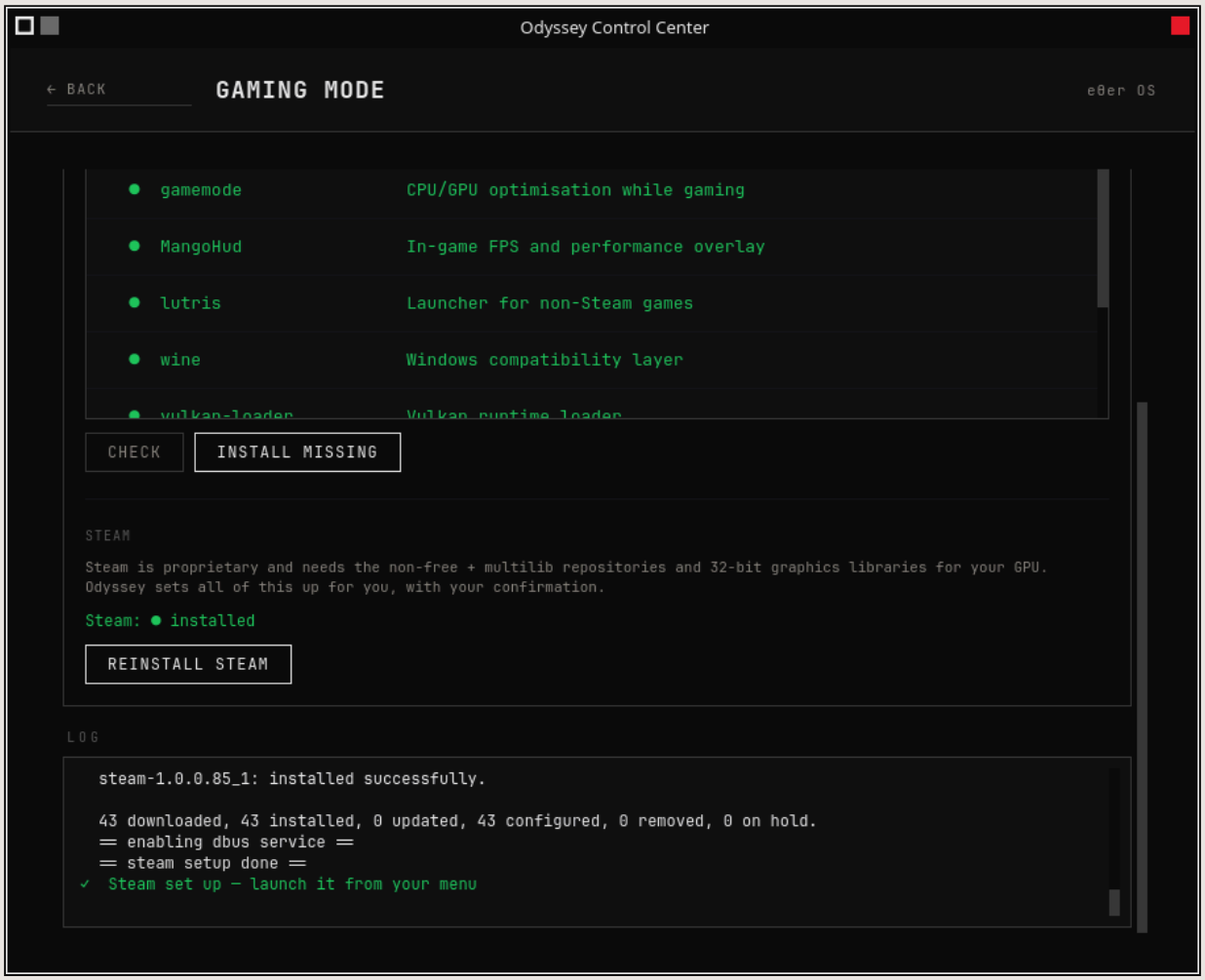

FIG · La tab PACKAGES dopo la configurazione: tutti i pacchetti verdi, Steam che mostra "● installed" con REINSTALL STEAM, e il log che termina "Steam set up — launch it from your menu."

WHAT	Gaming Mode → tab PACKAGES → SET UP STEAM → rivedi il dialogo → Sì.
WHY	Steam è proprietario e ha bisogno dei repository non-free/multilib più librerie a 32-bit specifiche per la GPU che Void non abilita per default — ottenere questo correttamente a mano è un ostacolo comune per chi è alle prime armi con un sistema basato su Void.
RESULT	I repository necessari, Steam stesso, e le librerie grafiche a 32-bit corrispondenti alla tua GPU specifica, tutti installati — avvia Steam dal tuo menù applicazioni dopo.
RISK	Basso — questo abilita solo repository e aggiunge pacchetti; niente di esistente viene rimosso o riconfigurato. Il dialogo mostra l'elenco completo di cosa succederà prima che tu confermi.

10.3 GameMode e MangoHud

GameMode (la propria tab) è il demone di prestazioni per singolo gioco di Feral Interactive — un gioco lanciato attraverso di esso richiede un boost temporaneo di prestazioni (governor CPU, priorità I/O) solo per il proprio processo, restituito automaticamente quando il gioco si chiude. **MangoHud** (la propria tab) è l'overlay a schermo che mostra FPS, frame time, carico CPU/GPU e temperature durante il gioco. Entrambe le tab offrono controlli di installazione/verifica e, per MangoHud, configurazione rapida dell'aspetto dell'overlay.

10.4 Proton

Gestione del layer di compatibilità per eseguire giochi Windows attraverso Steam — verificando cosa è installato e disponibile, e installando versioni aggiuntive di Proton, incluse build della community come Proton-GE, quando un gioco specifico ne richiede una che Valve non distribuisce per default.

```
▸ ADVANCED

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

10.5 Advanced

COSA SCRIVE ACTIVATE

Gli stessi file sottostanti delle azioni MAKE PERSISTENT dei pannelli Kernel e Memory — Gaming Mode è un pacchetto, non un meccanismo separato:

```
# valori sysctl resi persistenti
$ cat /etc/sysctl.d/99-odyssey.conf
# CPU governor — non è sysctl, è basato su sysfs, non reso persistente
# da questo meccanismo; riapplica dopo il boot se ci fai affidamento
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

COSA ESEGUE DAVVERO SET UP STEAM

Il dialogo di conferma stesso è quasi una divulgazione tecnica completa, ma concretamente, su un sistema 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
# i sistemi NVIDIA scaricano in aggiunta nvidia-libs-32bit invece di mesa-dri-32bit —
# il pannello rileva il vendor della tua GPU e adatta di conseguenza la lista pacchetti
```

GAMEMODE E MANGOHUD A MANO

```
# lancia qualunque gioco con GameMode attivo
$ gamemoderun ./game_binary
# oppure, per un gioco Steam, opzioni di avvio:
$ gamemoderun %command%
# overlay MangoHud, opzioni di avvio:
$ mangohud %command%
# il file di config proprio di MangoHud, per regolare cosa mostra l'overlay
$ $EDITOR  /.config/MangoHud/MangoHud.conf
```

INSTALLARE UNA VERSIONE SPECIFICA DI PROTON-GE

La tab PROTON avvolge `protonplus` o un gestore di versioni equivalente — l'equivalente manuale è scaricare una release da `GloriousEggroll/proton-ge-custom` su GitHub ed estrarla nella directory degli strumenti di compatibilità di Steam:

```
$ mkdir -p  /.steam/root/compatibilitytools.d
$ tar -xf GE-Proton-VERSION.tar.gz -C  /.steam/root/compatibilitytools.d
# poi selezionalo per singolo gioco in Steam: Properties → Compatibility
```

CHAPTER 11

Telemetry

Nonostante il nome, questo pannello non è un'impostazione da configurare — è un monitor in tempo reale. Odyssey non raccoglie né invia nulla su di te, quindi non c'è niente da disattivare qui. Quello che questo pannello fa invece è mostrarti, in tempo reale, ogni connessione in uscita che il tuo **stesso** sistema sta facendo — così "con cosa sta parlando la mia macchina in questo momento" è qualcosa a cui puoi rispondere guardando, non fidandoti di un'affermazione. Si chiama **Reverse Telemetry** proprio per questo: capovolge l'osservazione, puntandola sul sistema stesso, per conto tuo.

11.1 Guida visiva

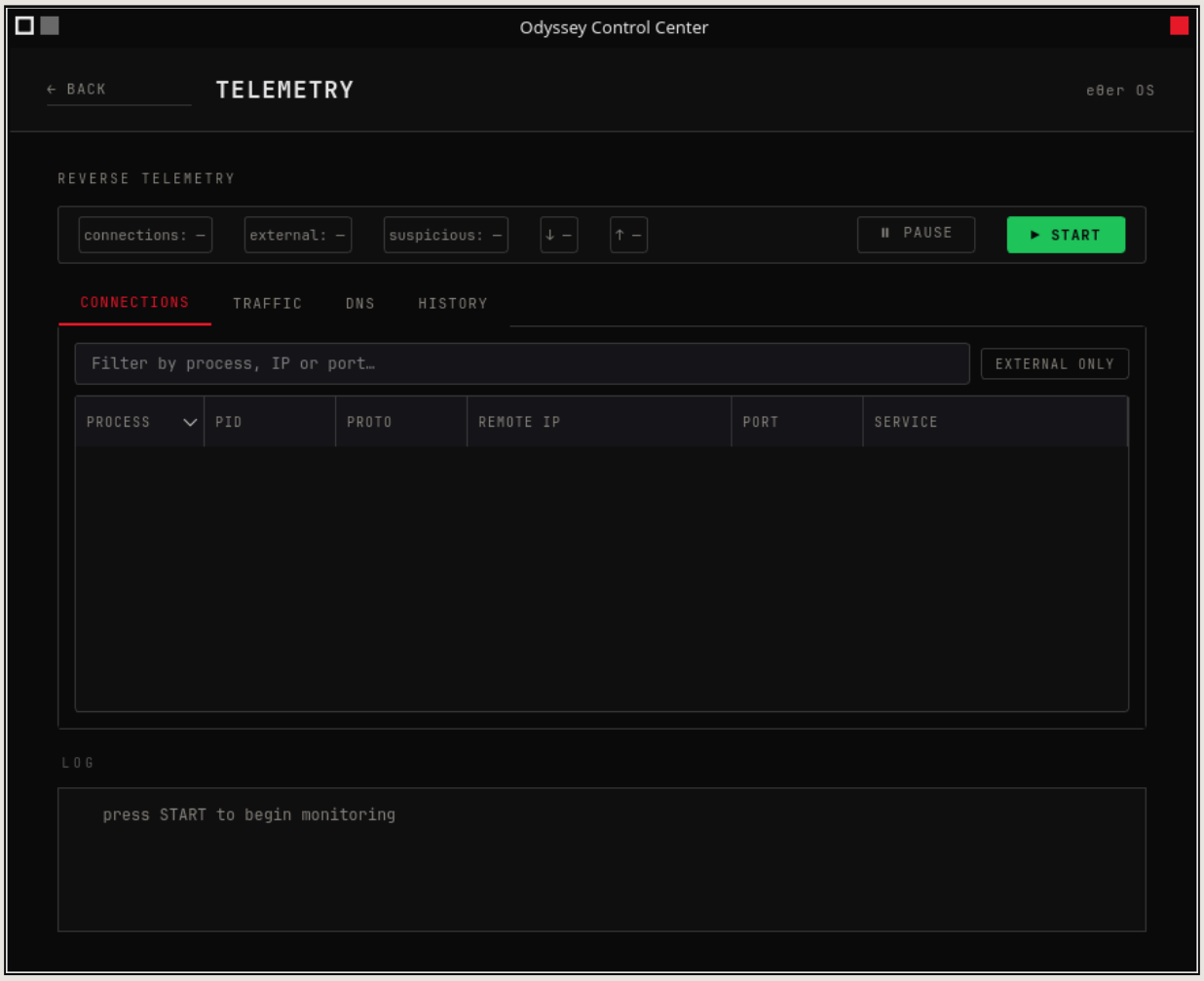

FIG · Prima di avviare: connections/external/suspicious che mostrano tutti "–", un pulsante verde START, e il log che legge "press START to begin monitoring." Le quattro tab — CONNECTIONS, TRAFFIC, DNS, HISTORY — restano vuote sopra una tabella non popolata.

1 START

Avvia il monitoraggio in tempo reale — la barra di stato e la tab attiva si popolano immediatamente.

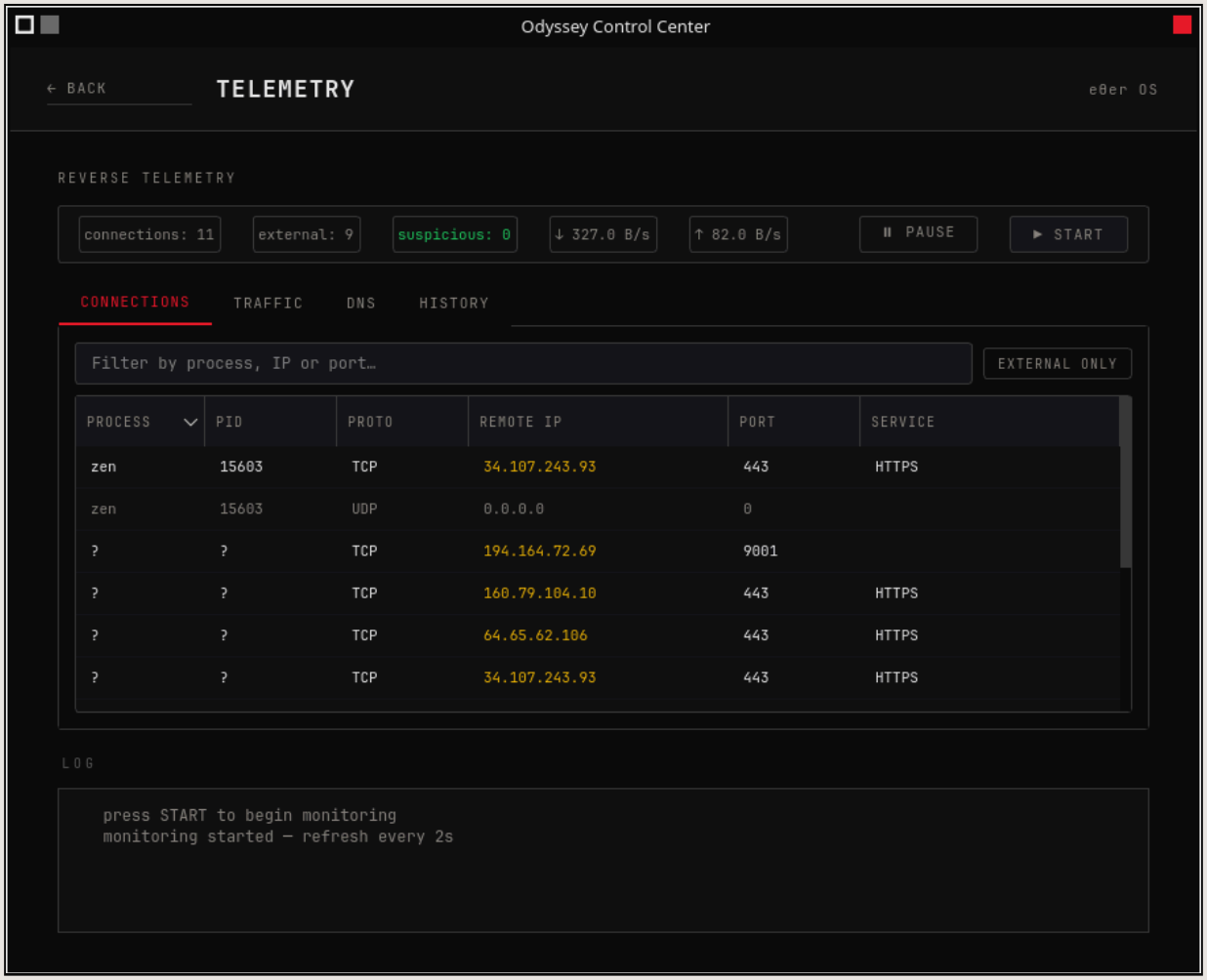

FIG · Dopo l'avvio: "connections: 11 · external: 9 · suspicious: 0 · ↓327.0 B/s · ↑82.0 B/s," PAUSE ora disponibile, e la tabella CONNECTIONS piena di righe reali: processo "zen" su TCP verso 34.107.243.93:443 (HTTPS), diverse righe contrassegnate "?" per processo/PID con connessioni verso altri indirizzi.

La barra di stato in cima fornisce totali in tempo reale — connessioni, esterne (vs. locali), sospette, velocità di download/upload attuali — e **PAUSE** congela la vista senza perdere ciò che è stato catturato, **START** la riprende.

LE QUATTRO TAB

TAB	COSA MOSTRA
CONNECTIONS	Una riga per connessione attiva: processo, PID, protocollo, indirizzo remoto, porta, e il servizio noto per quella porta (443 → HTTPS, 22 → SSH). Ordinabile per qualunque colonna.
TRAFFIC	Throughput per interfaccia — velocità di download/upload in questo momento, più totali cumulativi da quando è iniziato il monitoraggio.
DNS	Un flusso in tempo reale delle risoluzioni di nomi a dominio man mano che avvengono — spesso più chiaro di un semplice IP, perché un hostname ti dice **cosa**, non solo **dove**.
HISTORY	Tutto ciò che la sessione ha catturato, come log scorrevole — utile per individuare qualcosa di breve che non è più una connessione attiva nel momento in cui guardi.

Le righe dove il processo mostra `?` (visibili nello screenshot) significano che la connessione è stata catturata ma il processo d'origine non è stato risolvibile dal PID in quell'istante — comune per connessioni molto brevi. Un campo **filtro** sopra la tabella restringe per processo, IP o porta, ed **EXTERNAL ONLY** nasconde il traffico puramente locale.

WHAT	Telemetry → START → osserva CONNECTIONS, oppure ordina per PROCESS o REMOTE IP per indagare su qualcosa di specifico.
WHY	È la risposta diretta e concreta a "c'è qualcosa su questa macchina che raggiunge la rete che non dovrebbe" — letta direttamente dalle tabelle di connessione del kernel stesso, non dall'affermazione di un fornitore sul proprio software.
RESULT	Un quadro live e accurato di ogni connessione in uscita, aggiornato continuamente mentre è in esecuzione e non in pausa.
RISK	Nessuno — interamente in sola lettura.

Le connessioni da processi senza un motivo ordinario per raggiungere la rete da soli — una shell nuda, un interprete di script in esecuzione autonoma — vengono contrassegnate automaticamente sotto "suspicious" nella barra di stato, così un'anomalia risalta invece di restare sepolta tra il traffico normale del tuo browser e dei servizi di sistema.

▸ ADVANCED

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

11.2 Advanced

DA DOVE VENGONO I DATI

Il pannello legge le stesse tabelle di connessione a livello kernel che leggerebbe qualunque strumento standard, aggiornate a un intervallo breve mentre il monitoraggio è attivo — nessun driver o modulo kernel speciale proprio. Gli equivalenti da riga di comando:

```
# ogni socket aperto, con il processo proprietario — stessi campi di CONNECTIONS
$ sudo ss -tupn
# throughput per interfaccia — stessi numeri di TRAFFIC
$ ip -s link
# una vista live one-shot, se la vuoi fuori dal pannello
$ sudo nethogs
```

RISOLUZIONI DNS, A MANO

```
# cattura live delle query DNS in uscita (serve uno strumento di packet capture)
$ sudo tcpdump -i any -n port 53
```

PERCHÉ ALCUNE RIGHE MOSTRANO PROCESS COME "?"

Abbinare una connessione live al processo che la possiede richiede leggere `/proc/PID/fd` per ogni processo candidato nel momento della cattura — se la connessione viene aperta e chiusa più velocemente dell'intervallo di aggiornamento del pannello, o il processo proprietario è già terminato, il

pannello non ha la finestra per fare quell'abbinamento e mostra ? invece di indovinare. È un limite di tempistica, non un segno che qualcosa venga nascosto.

CHAPTER 12

About

Un'unica scheda diagnostica esaustiva della macchina, aggiornata ogni volta che apri la tab, così riflette sempre ciò che è cambiato negli altri pannelli senza dover riaprire il Control Center.

12.1 Guida visiva

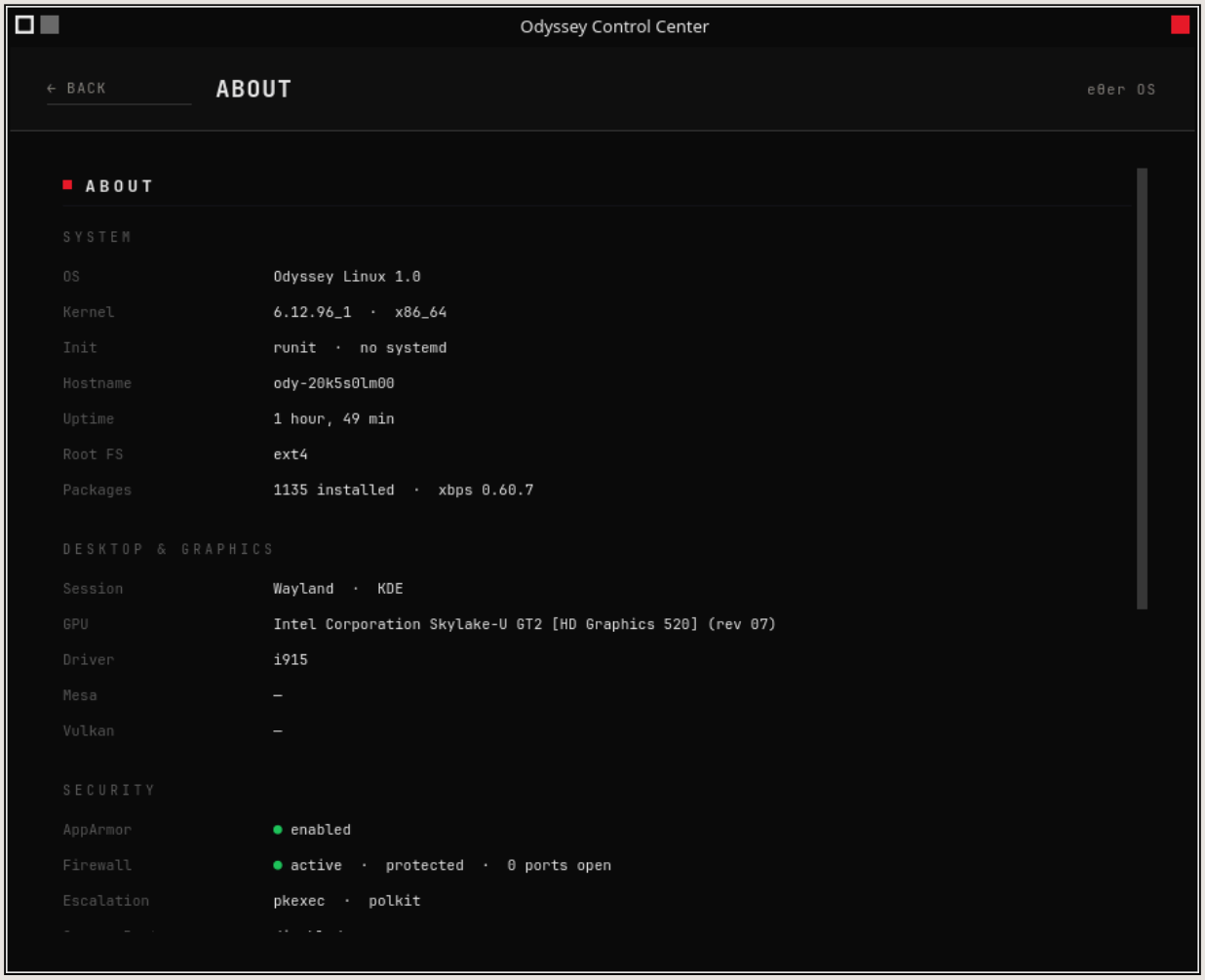

FIG · Il pannello scorso fino in cima: sezione SYSTEM che mostra OS "Odyssey Linux 1.0," Kernel 6.12.96_1 · x86_64, Init runit · no systemd, Hostname, Uptime, Root FS ext4, Packages "1135 installed · xbps 0.60.7"; DESKTOP & GRAPHICS sotto con Session, GPU, Driver, Mesa, Vulkan; SECURITY che inizia a mostrare AppArmor "● enabled" e Firewall "● active · protected · 0 ports open."

Tre sezioni, in ordine: **SYSTEM** (kernel, init, hostname, uptime, filesystem, conteggio pacchetti), **DESKTOP & GRAPHICS** (tipo di sessione, GPU, driver, supporto Mesa/Vulkan), e più sotto **SECURITY** (stato di AppArmor e Firewall con indicatori colorati, non solo testo — verde per abilitato/attivo) e **HARDWARE** (scheda madre, audio).

WHAT	About → scorri le tre sezioni per controllare qualunque valore — versione del kernel, driver in uso, se AppArmor e il firewall sono davvero attivi.
WHY	È il modo più veloce per confermare lo stato dell'intero sistema in un unico posto, senza aprire cinque pannelli diversi.
RESULT	Un'istantanea completa e aggiornata — rinfrescata a ogni visita alla tab, così non mostra mai informazioni obsolete di una sessione precedente.
RISK	Nessuno — interamente in sola lettura.

12.2 Copy report

Un solo pulsante in cima al pannello, **COPY REPORT**, mette l'intera scheda negli appunti come testo semplice — ogni sezione, ogni valore — pronta da incollare direttamente in un post del forum o in una segnalazione sul bug tracker.

WHAT	About → COPY REPORT → incolla nella tua segnalazione di bug o post del forum.
WHY	Le segnalazioni di bug sono utili solo quanto i dettagli di sistema allegati, e digitare a mano versione del kernel, driver, stato di AppArmor e una dozzina di altri campi è esattamente il tipo di attrito che fa saltare il passaggio a molte persone.
RESULT	Un report di sistema completo e correttamente formattato negli appunti, pronto da incollare — le stesse informazioni che un maintainer altrimenti dovrebbe chiederti campo per campo.
RISK	Nessuno per il tuo sistema. Il report è informazione diagnostica (versioni, stati, modello hardware) — non include file personali o credenziali — ma come con qualunque dump diagnostico, rileggilo prima di pubblicarlo da qualche parte in pubblico se non sei sicuro di cosa contenga.

▸ ADVANCED

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

12.3 Advanced

DA DOVE VIENE OGNI CAMPO

Niente in questo pannello è inventato o messo in cache dal momento dell'installazione — ogni valore viene interrogato dal vivo, ogni volta che la tab si apre:

```
$ 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
```

Sezione Security nello specifico:

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

PERCHÉ CONVIENE CONTROLLARE ABOUT PER PRIMO

Poiché interroga il sistema di nuovo a ogni visita, About è il modo più veloce per confermare che una modifica fatta in un altro pannello abbia davvero avuto effetto — applica una modalità del firewall in Firewall, poi controlla la sezione Security di About invece di riaprire Firewall stesso. È costruito per essere il controllo incrociato “ha davvero funzionato” per il resto del Control Center.

CHAPTER 13

NVIDIA driver

> **IN ATTESA DI INTEGRAZIONE**
>
> Questo capitolo è un segnaposto. Il modulo NVIDIA compare nel Control Center solo su macchine con hardware NVIDIA rilevato, e gli screenshot per esso possono essere catturati solo su una macchina di questo tipo — verranno forniti separatamente e questo capitolo completato in una futura revisione di questa guida. È tenuto per ultimo, dopo About, apposta perché aggiungerlo in seguito non sposti la numerazione delle pagine o i riferimenti alle sezioni di nessun altro capitolo.

Cosa si sa già e ancorerà il capitolo completo: il pannello porta un badge **BETA** visibile e una nota iniziale che il driver stesso è proprietario — pacchettizzato da Void a partire dall'installer ufficiale di NVIDIA con un checksum fissato, ma non verificabile in modo indipendente, la stessa onestà che il resto di Odyssey applica a sé stesso. Rileva la tua GPU specifica confrontandola con un database integrato e sceglie automaticamente il branch driver corretto (current/Turing-e-più-recenti, oppure uno dei tre branch legacy per schede più vecchie, ripiegando sul driver open-source `nouveau` quando nulla di proprietario si applica). L'installazione gira come una pipeline tracciata in sette step con un tracker di avanzamento visibile, e il pannello stampa il proprio comando di recupero in anticipo — `odyssey-nvidia --revert` da una console testuale — prima ancora che tu clicchi installa. Sui laptop ibridi, una sezione PRIME render-offload compare automaticamente, permettendo a un gioco o programma specifico di girare sulla GPU NVIDIA mentre tutto il resto resta sulla GPU integrata per l'autonomia della batteria.