ODYSSEY LINUX · संदर्भ दस्तावेज़ीकरण

Odyssey Control Center — व्यावहारिक गाइड

हर पैनल, सेक्शन दर सेक्शन। वह क्या करता है, क्या बदलता है,
और कहाँ क्लिक करना है — उनके लिए लिखा गया है जिन्होंने इसे पहले कभी इस्तेमाल नहीं किया।

Launch Edition

Don’t trust me. Verify me.

— nobody

Odyssey Linux — Odyssey Control Center: व्यावहारिक गाइड
Launch Edition। इसमें Control Center को उसी रूप में शामिल किया गया है जैसे वह Launch Edition ISO पर दिया जाता है — Boot से लेकर About तक दस पूरे पैनल, साथ ही NVIDIA ड्राइवर मॉड्यूल के लिए एक अस्थायी अध्याय (जो केवल NVIDIA हार्डवेयर पर दिखाई देता है, और उस हार्डवेयर वाले किसी उपयोगकर्ता के स्क्रीनशॉट का इंतज़ार कर रहा है)।

Copyright © 2026 the Odyssey Linux project.
Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) लाइसेंस के तहत प्रकाशित।

यह गाइड Security and Hardening Guide का व्यावहारिक साथी है, जो Firewall और AppArmor पैनल को पूरी तकनीकी गहराई के साथ कवर करता है — यहाँ आपको चरण-दर-चरण दृश्य गाइड और बाकी के लिए एक क्रॉस-रेफरेंस मिलेगा।

यहाँ बताई गई हर स्क्रीन, बटन और व्यवहार को Control Center के असली सोर्स कोड और असली स्क्रीनशॉट से पढ़ा गया है, याद्दाश्त से नहीं।

बटन लेबल, पैनल/टैब के नाम, फ़ाइल पथ और कमांड अंग्रेज़ी में ही रहते हैं क्योंकि वही प्रोग्राम का असली इंटरफ़ेस है: उन्हें Control Center में अनुवादित रूप में खोजने से वे नहीं मिलेंगे।

विषय-सूची

1 Control Center क्या है 1
1.1 वे पैटर्न जो आप हर पैनल में देखेंगे 2
2 Boot 3
2.1 दृश्य गाइड — दूसरा ऑपरेटिंग सिस्टम खोजना और डिफ़ॉल्ट सेट करना 3
2.2 दृश्य गाइड — डिफ़ॉल्ट रूप से क्या बूट होगा, यह चुनना 5
2.3 दृश्य गाइड — EFI टैब 7
2.4 कर्नेल पैरामीटर 9
2.5 Advanced 9
3 Kernel 11
3.1 दृश्य गाइड — प्रीसेट 11
3.2 दृश्य गाइड — SYSCTL, समूह-दर-समूह 12
3.3 दृश्य गाइड — CPU governor 14
3.4 दृश्य गाइड — hugepages 15
3.5 दृश्य गाइड — modules 16
3.6 Advanced 17
4 Memory & Swap 19
4.1 दृश्य गाइड 19
4.2 Advanced 20
5 Firewall 22
5.1 दृश्य गाइड 22
5.2 Advanced 23
6 AppArmor 24
6.1 दृश्य गाइड — STATUS 24
6.2 दृश्य गाइड — PROFILES 25
6.3 दृश्य गाइड — COMMUNITY 26
6.4 दृश्य गाइड — VIOLATIONS 27
6.5 दृश्य गाइड — EDITOR 28
6.6 Advanced 29
7 Privacy 31
7.1 दृश्य गाइड — SearXNG, चरण-दर-चरण 31
7.2 दृश्य गाइड — Tor, चरण-दर-चरण 33
7.3 दृश्य गाइड — दोनों को मिलाना 36
7.4 Advanced 37
8 Containers 39
8.1 दृश्य गाइड — इंजन 39
8.2 दृश्य गाइड — rootless बनाम rootful, और ख़ाली हालत 41
8.3 दृश्य गाइड — चुनिंदा कैटलॉग, शुरू से अंत तक 41
8.4 दृश्य गाइड — कंटेनर सूची और उसकी कार्रवाइयाँ 45
8.5 दृश्य गाइड — एक कंटेनर चलाना, और उसे खोलना 46
8.6 डिस्क इस्तेमाल 47
8.7 Advanced 47
9 Gaming Mode 49
9.1 दृश्य गाइड — मुख्य स्विच 49
9.2 दृश्य गाइड — Packages और Steam 50
9.3 GameMode और MangoHud 52
9.4 Proton 53
9.5 Advanced 53
10 Telemetry 55
10.1 दृश्य गाइड 55
10.2 Advanced 57
11 About 58

11.1 दृश्य गाइड 58
11.2 Copy report 59
11.3 Advanced 59
12 NVIDIA driver 61

CHAPTER 1

Control Center क्या है

Odyssey Control Center एक ही लॉन्चर के पीछे दस छोटे, स्वतंत्र प्रोग्राम हैं — सिस्टम के हर हिस्से के लिए एक: Boot, Kernel, Memory & Swap, Firewall, AppArmor, Privacy, Containers, Gaming Mode, Telemetry, About, और (NVIDIA हार्डवेयर पर) NVIDIA ड्राइवर मॉड्यूल।

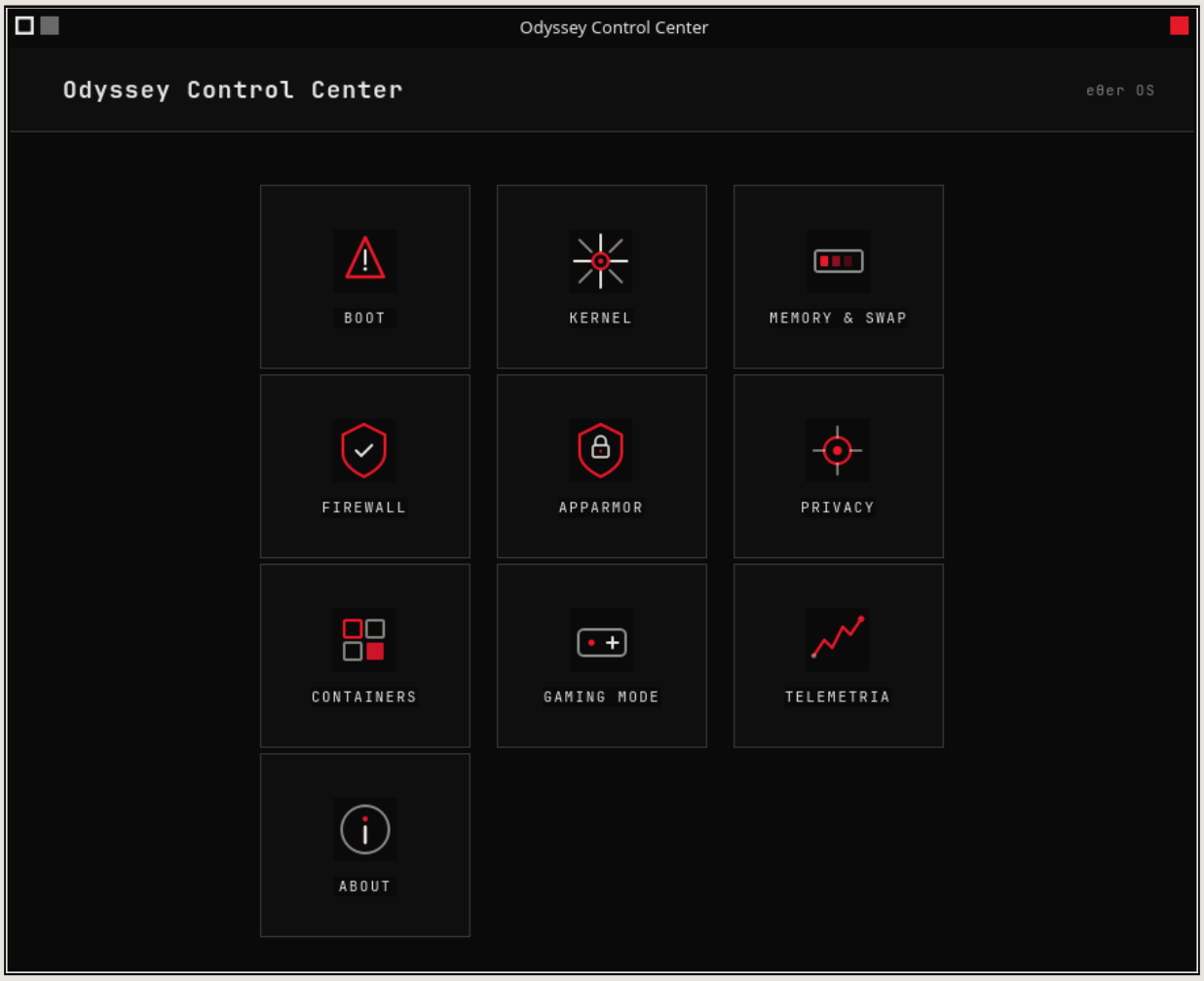

FIG · Control Center की होम स्क्रीन — हर पैनल के लिए एक टाइल।

हर पैनल नीचे मौजूद असली Linux तंत्र के ऊपर एक पतली ग्राफ़िकल परत होती है: GRUB की कॉन्फ़िगरेशन, कर्नेल का अपना sysctl इंटरफ़ेस, `nftables`, `podman`। कोई पैनल कभी अपनी खुद की निजी स्थिति नहीं गढ़ता — वह असली फ़ाइलें और कमांड पढ़ता है, उन्हें आपको दिखाता है, और वही लिखता है जो आपने माँगा था। यह जो कुछ भी करता है वह छिपा हुआ नहीं है, और यह जो कुछ भी करता है वह टर्मिनल से हाथ से भी किया जा सकता है, उन्हीं सटीक कमांड के साथ जो यह गाइड हर बटन के बगल में दिखाती है।

यह एक व्यावहारिक गाइड है, हर विकल्प की सूची नहीं — यह इस सवाल का जवाब देती है कि “मैं क्या क्लिक करूँ, और अगर मैं करूँ तो क्या होगा”। हर अध्याय की संरचना एक जैसी है, दो हिस्सों में: पहले एक **दृश्य गाइड**, स्क्रीनशॉट-दर-स्क्रीनशॉट, बटन-दर-बटन, उनके लिए लिखी गई जिन्होंने Control Center कभी खोला ही नहीं है — फिर एक काले बैनर से चिह्नित **ADVANCED** सेक्शन, जो विस्तार में जाता है: फ़ाइलों के सटीक पथ, हर बटन नीचे जो असली कमांड चलाता है, और अगर आप पैनल को पूरी तरह छोड़ना चाहें तो टर्मिनल से क्या करना है। आप सिर्फ़ दृश्य गाइड पढ़ सकते हैं, या अगर आप पहले से Linux जानते हैं तो सीधे Advanced पर जा सकते हैं — दोनों अपने आप में पूरे हैं।

दो पैनल जिनमें सुरक्षा के असली जोखिम हैं — Firewall और AppArmor — उनके लिए गहराई से जानकारी **Security and Hardening Guide** में है; यहाँ आपको दृश्य गाइड, ज़रूरी बातें, और एक क्रॉस-रेफरेंस मिलेगा।

1.1 वे पैटर्न जो आप हर पैनल में देखेंगे

कुछ नियम सभी दस पैनल में दोहराए जाते हैं — इन्हें एक बार सीखना सब पर लागू होता है।

पासवर्ड का अनुरोध, हर पैनल में सिर्फ़ एक बार

अधिकतर पैनल को root के मालिकाना हक़ वाली फ़ाइलें पढ़नी या लिखनी पड़ती हैं — GRUB की कॉन्फ़िगरेशन, sysctl, फ़ायरवॉल का नियम-सेट। जब पहली बार किसी पैनल को इसकी ज़रूरत पड़ती है, तो आपको एक ही प्रमाणीकरण डायलॉग मिलता है। पैनल इस तरह बनाए गए हैं कि वे अपनी पढ़ने की क्रियाओं को एक साथ जोड़ दें: उदाहरण के लिए, Boot को खोलना GRUB की कॉन्फ़िगरेशन **और** EFI फ़र्मवेयर की स्थिति दोनों को एक ही अनुरोध में पढ़ लेता है, इसलिए आप अपना पासवर्ड एक बार टाइप करते हैं, दो बार नहीं। उसी सत्र में किसी पैनल को दोबारा खोलने पर वह दोबारा नहीं पूछता; उसके बाद की क्रियाएँ (बदलाव लागू करना, फिर से लोड करना) पैनल के अपने लॉग में अपनी प्रगति दिखाती हैं।

स्टेप ट्रैकर

जो पैनल किसी चीज़ के इस्तेमाल लायक बनने से पहले उसकी इंस्टॉलेशन में आपका मार्गदर्शन करते हैं — Boot में os-prober, Privacy में SearXNG और Tor, Containers में कंटेनर इंजन — वे ऊपर एक छोटा नंबर वाला ट्रैकर दिखाते हैं: कदमों की एक कतार, जिसमें मौजूदा कदम उभरा हुआ दिखता है, और सिर्फ़ एक ऐक्शन बटन जो हमेशा **अगला कदम** करता है, चाहे वह जो भी हो। आपको कभी अंदाज़ा नहीं लगाना पड़ता कि पहले क्या क्लिक करें।

नीचे का लॉग

हर पैनल एक छोटे, स्क्रॉल होने वाले लॉग पर ख़त्म होता है। यह सजावट नहीं है — यह वह सटीक कमांड है जो चलाया गया, उसका आउटपुट, और एक रंगीन फ़ैसला (सफलता के लिए हरा, असफलता के लिए लाल)। अगर कोई पैनल कुछ ऐसा करता है जो आपको समझ नहीं आता, तो असली जवाब लॉग में ही मिलता है।

TRY VS. MAKE PERSISTENT

बटनों की एक जोड़ी बार-बार दिखती है, ख़ासकर Kernel और Memory में: **TRY NOW** (या **APPLY NOW**) चालू सिस्टम को तुरंत बदल देता है, और अगर आप कुछ और नहीं करते तो अगले रीबूट पर वापस पहले जैसा हो जाता है। **MAKE PERSISTENT** वही मान `/etc/` के अंदर एक कॉन्फ़िगरेशन फ़ाइल में लिख देता है, ताकि वह रीबूट के बाद भी बना रहे। यह अलगाव जानबूझकर है — किसी मान को आज़माने में कोई लागत नहीं है, और जब तक आप इसे दो बार न कहें, कुछ भी स्थायी नहीं है।

CHAPTER 2

Boot

Boot पैनल में हर उस मशीन पर दो टैब होते हैं जो UEFI मोड में बूट होती है: **GRUB** (बूट मेन्यू ख़ुद) और **EFI** (उसके नीचे का फ़र्मवेयर स्तर)। पुरानी मशीनों पर, जिनमें सिर्फ़ BIOS है, सिर्फ़ GRUB होता है, क्योंकि नीचे EFI स्तर बिल्कुल मौजूद ही नहीं होता।

दो स्तर, एक नहीं

GRUB वह बूट मेन्यू है जो आप स्क्रीन पर देखते हैं — यह तय करता है कि कौन-सा ऑपरेटिंग सिस्टम शुरू होगा, जब GRUB ख़ुद पहले से ही चल रहा हो। **EFI/UEFI फ़र्मवेयर** GRUB से एक स्तर नीचे होता है — यह तय करता है कि मशीन सबसे पहले नियंत्रण किस बूटलोडर को देती है, यह तो GRUB के लोड होने से भी पहले की बात है। ज़्यादातर लोग EFI टैब को कभी छूते ही नहीं; यह उन मामलों के लिए है जहाँ GRUB नहीं पहुँच सकता, जैसे कोई दूसरी Windows इंस्टॉलेशन जो अपनी बूट एंट्री ख़ुद, स्वतंत्र रूप से संभालती है।

2.1 दृश्य गाइड — दूसरा ऑपरेटिंग सिस्टम खोजना और डिफ़ॉल्ट सेट करना

GRUB टैब **Other OS detection** सेक्शन में खुलता है। Odyssey डिफ़ॉल्ट रूप से दूसरे सिस्टम को स्कैन नहीं करता — ज़्यादातर इंस्टॉलेशन में डिस्क पर सिर्फ़ Odyssey होता है, और हर बार बूट मेन्यू फिर से बनने पर स्कैन करना बेकार में समय बर्बाद करता है। इसे चालू करना तीन कदमों की एक गाइडेड प्रक्रिया है, जो एक छोटे नंबर वाले ट्रैकर के रूप में दिखाई जाती है।

1. **`Installed`**

   स्कैनिंग टूल, os-prober, अभी सिस्टम पर इंस्टॉल नहीं है। सिर्फ़ एक बटन: **INSTALL OS-PROBER**।

2. **`Enable scan`**

   टूल इंस्टॉल है, लेकिन GRUB अभी भी उसे नज़रअंदाज़ करने के लिए सेट है — यह Odyssey का डिफ़ॉल्ट है। सिर्फ़ एक बटन: **ENABLE SCAN**।

3. **`Scan`**

   अब GRUB इसका इस्तेमाल करेगा। सिर्फ़ एक बटन, जिसे आप भविष्य में जब भी कोई दूसरा सिस्टम जोड़ें या हटाएँ, तो फिर से इस्तेमाल करेंगे: **SCAN & REGENERATE**।

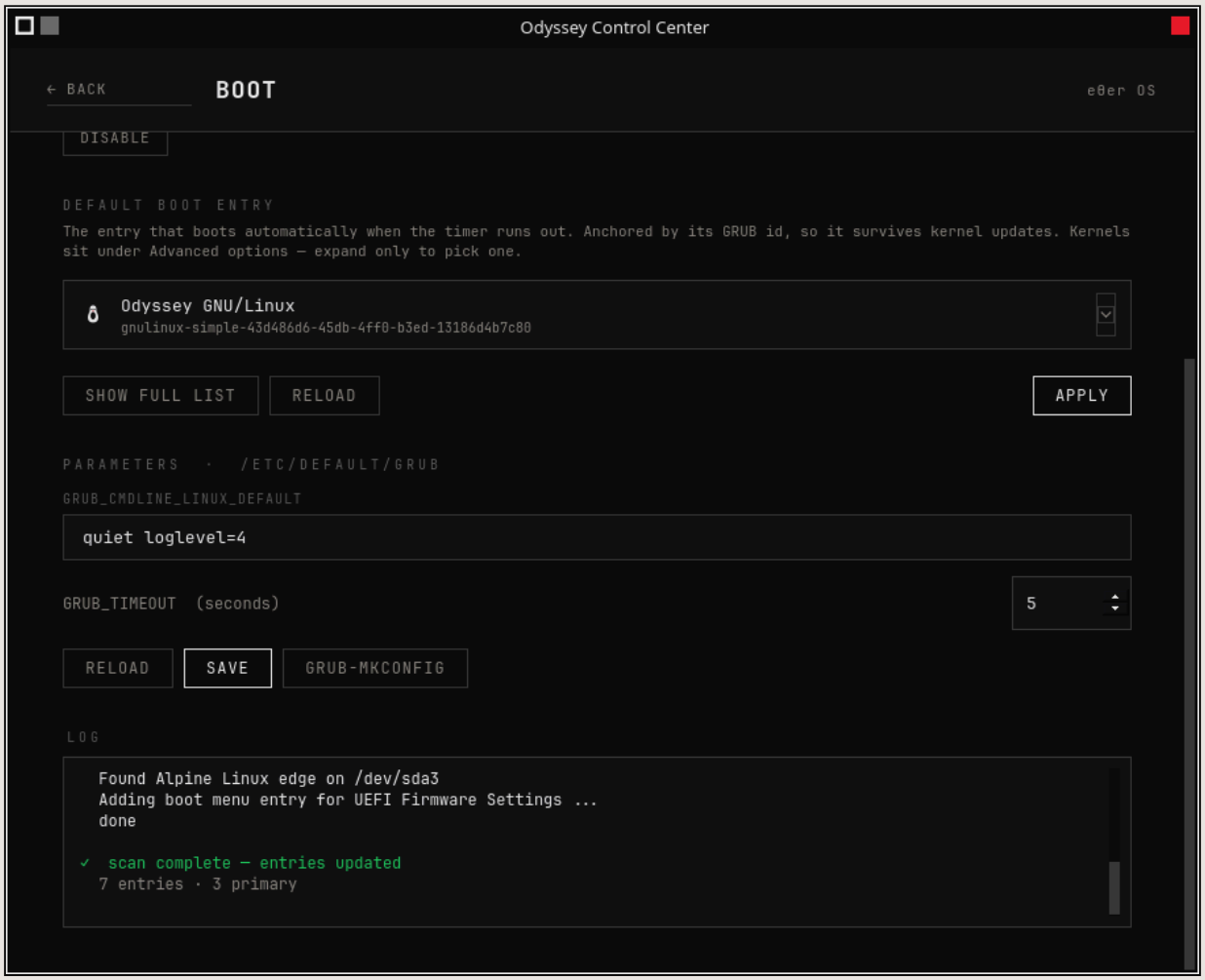

FIG · तीनों कदम हरे — os-prober इंस्टॉल हो चुका, स्कैन चालू है, तैयार। अकेला ऐक्शन बटन हमेशा मौजूदा कदम से मेल खाता है।

कदमों पर क्लिक करें (हर एक सिर्फ़ ज़रूरत पड़ने पर ही दोबारा प्रमाणीकरण माँगता है) और स्कैन चलाएँ। सेक्शन छोड़ने से पहले पैनल आपको बिल्कुल दिखाता है कि उसे क्या मिला:

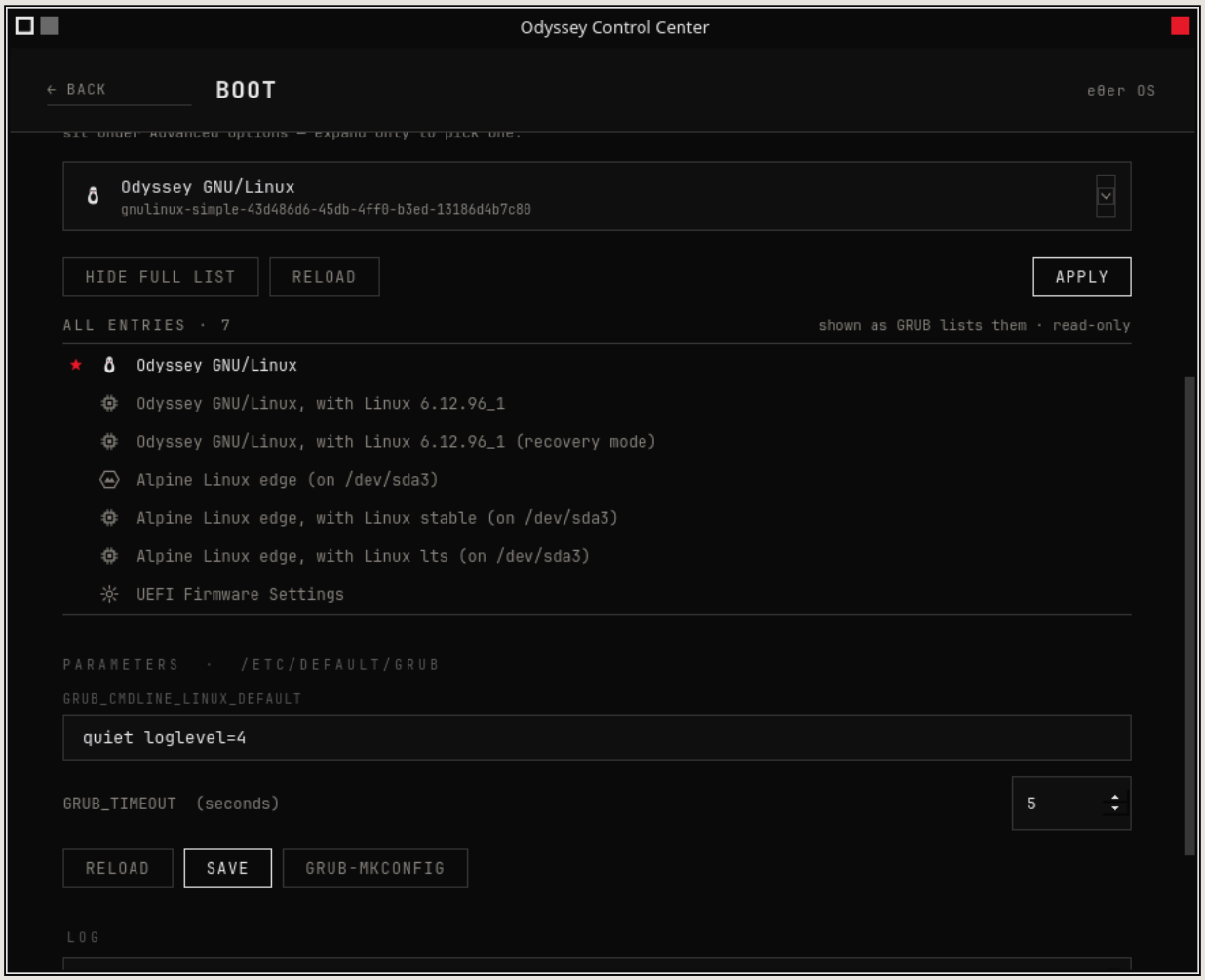

FIG · स्कैन पूरा: /dev/sda3 पर Alpine Linux मिला, Odyssey की अपनी एंट्रीज़ और UEFI Firmware Settings एंट्री के साथ — कुल 7 एंट्रीज़, 3 प्राथमिक।

लॉग इसे सरल शब्दों में पुष्टि करता है — **"scan complete — entries updated · 7 entries · 3 primary"**। अब os-prober को जो कुछ भी मिलता है, वह नीचे के सेक्शन में अपने आप दिखने लगता है, चुने जाने के लिए तैयार।

2.2 दृश्य गाइड — डिफ़ॉल्ट रूप से क्या बूट होगा, यह चुनना

Default boot entry तक स्क्रॉल करें। यह मौजूदा डिफ़ॉल्ट को एक लाइन में दिखाता है; उस पर क्लिक करने से एक ड्रॉपडाउन खुलता है जिसमें वह सब कुछ है जो आप उसकी जगह चुन सकते हैं।

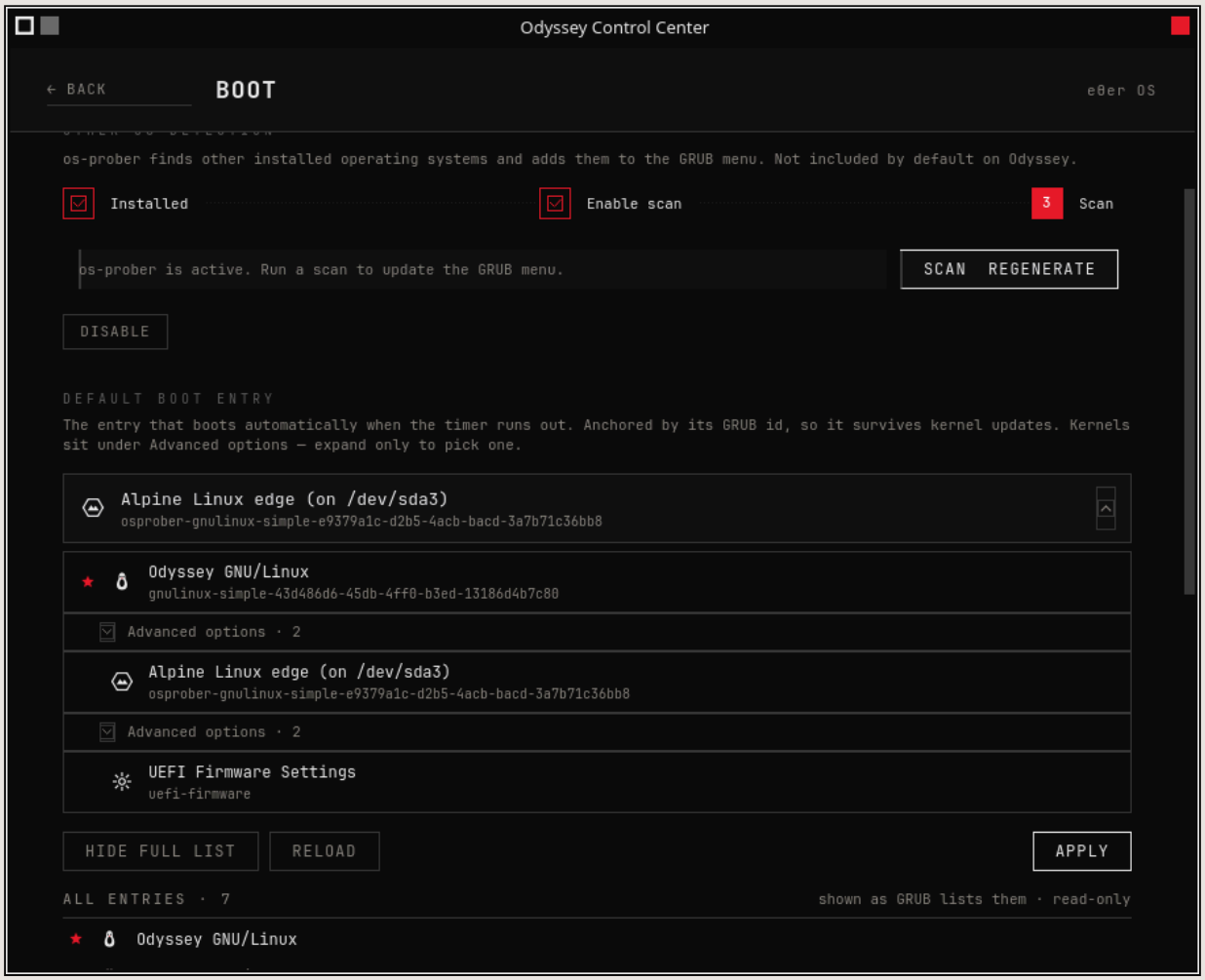

FIG · खुला हुआ ड्रॉपडाउन, जिसमें Alpine Linux को लंबित विकल्प के रूप में चुना गया है — अभी लागू नहीं हुआ।

सूची से कोई एंट्री चुनना उस चुनाव को **लंबित स्थिति में डाल देता है** — उसका अभी असर नहीं हुआ है। यह दिखता है कि यह सिर्फ़ लंबित है क्योंकि बॉक्स अभी भी नई चुनी हुई एंट्री दिखा रहा है लेकिन कुछ भी लिखा नहीं गया है। जो बटन इसकी पुष्टि करता है वह दाईं ओर **APPLY** है:

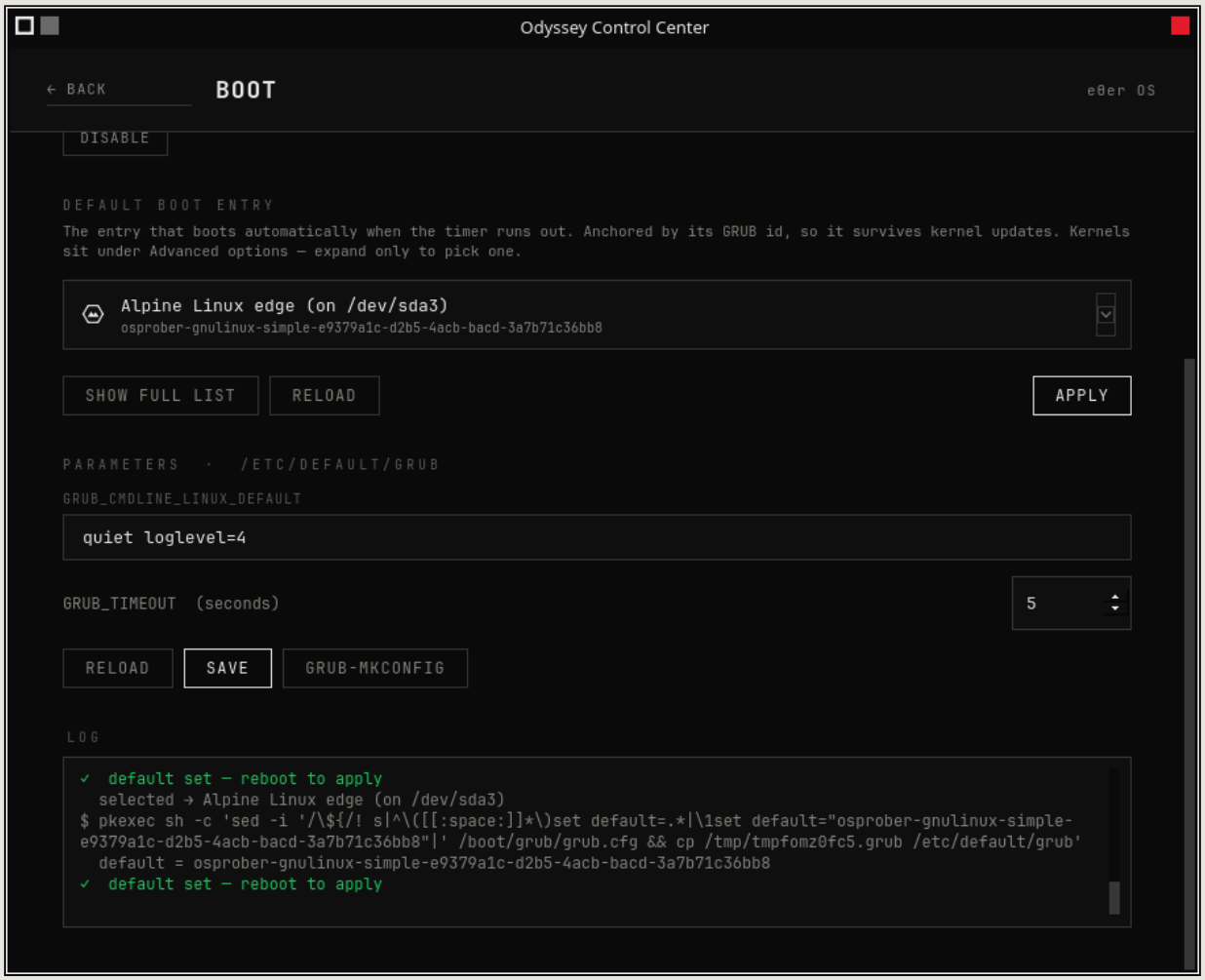

FIG · APPLY के बाद: लॉग पुष्टि करता है “default set — reboot to apply”, और सिलेक्टर अब Alpine Linux को चालू डिफ़ॉल्ट के रूप में दिखाता है।

WHAT	Boot → GRUB टैब → मौजूदा डिफ़ॉल्ट एंट्री पर क्लिक करें → ड्रॉपडाउन से एक अलग एंट्री चुनें → **APPLY** पर क्लिक करें।
WHY	आप डुअल-बूट कर रहे हैं और चाहते हैं कि कोई अलग सिस्टम अपने आप शुरू हो, या किसी कर्नेल अपडेट ने कोई पुराना कर्नेल चुना छोड़ दिया है और आप नवीनतम पर वापस जाना चाहते हैं।
RESULT	चुनी हुई एंट्री अगले रीबूट से अपने आप शुरू होती है। यह बदलाव उस एंट्री की स्थिर id से जुड़ा होता है, इसलिए यह भविष्य के कर्नेल अपडेट में भी बना रहता है।
RISK	आपके डेटा के लिए कोई नहीं — सबसे बुरी स्थिति में आप ग़लत एंट्री चुन लेते हैं और वापस आकर उसे फिर से बदल देते हैं।

SHOW FULL LIST हर एंट्री को बिल्कुल वैसे दिखाता है जैसे GRUB ख़ुद उन्हें सूचीबद्ध करता है — केवल पढ़ने के लिए, इस तरह समूहबद्ध कि अलग-अलग कर्नेल के वैरिएंट एक विस्तार योग्य “Advanced options” लाइन के नीचे रहें, न कि मुख्य सूची को भर दें। अगर आपको लगता है कि कॉन्फ़िगरेशन पैनल के बाहर बदल गई है, तो **RELOAD** GRUB की कॉन्फ़िगरेशन को फिर से पढ़ता है।

2.3 दृश्य गाइड — EFI टैब

ऊपर **EFI** टैब पर क्लिक करें। यह फ़र्मवेयर का स्तर है — वह जो आपके मदरबोर्ड का अपना बूट मेन्यू दिखाता है, इससे पहले कि कोई भी ऑपरेटिंग सिस्टम शुरू भी हुआ हो।

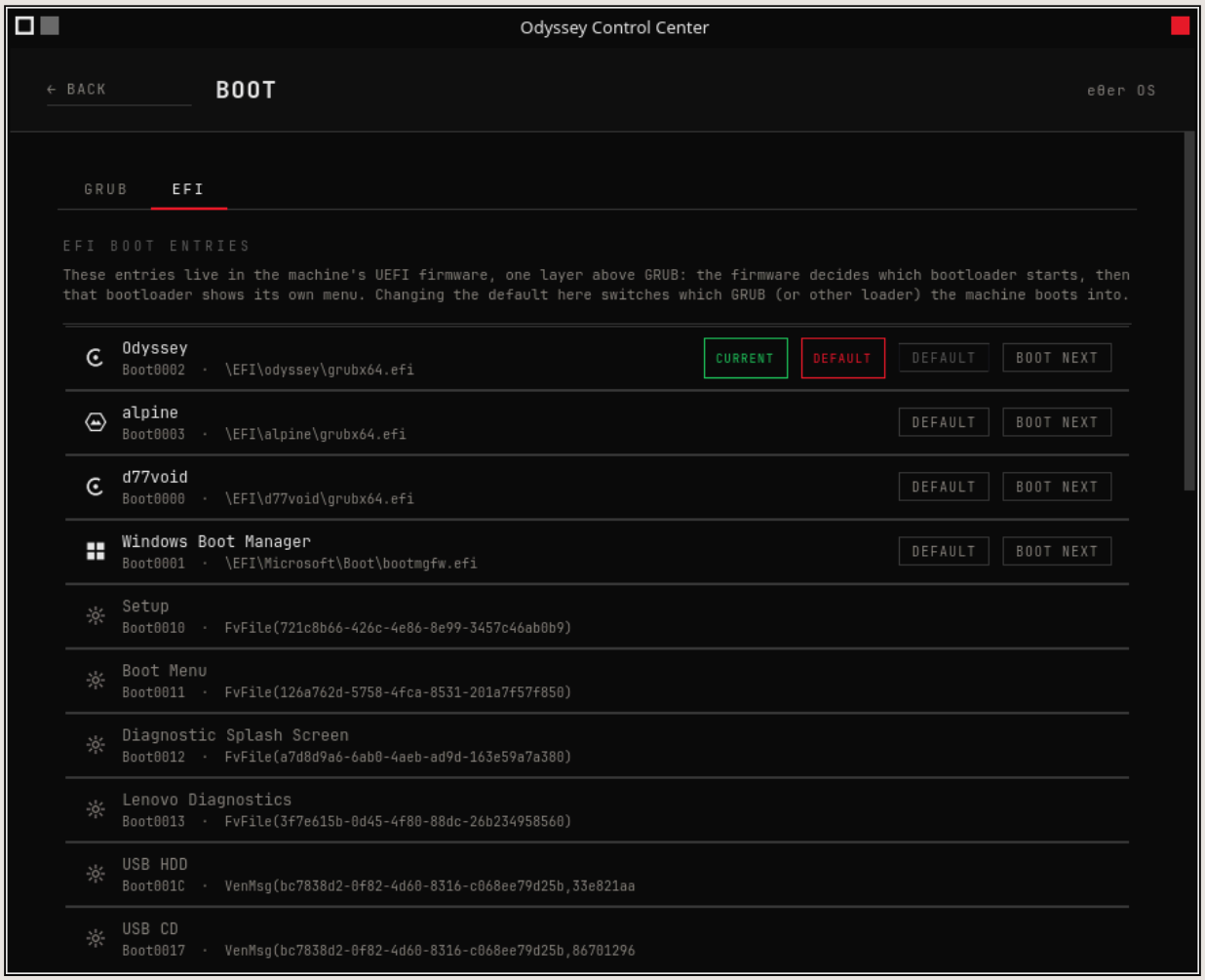

FIG · EFI टैब: Odyssey को हरे/लाल रंग में CURRENT और DEFAULT के रूप में चिह्नित किया गया है, alpine, d77void और Windows Boot Manager, हर एक के अपने DEFAULT / BOOT NEXT बटन के साथ, और नीचे फ़र्मवेयर यूटिलिटी एंट्रीज़।

हर लाइन एक फ़र्मवेयर एंट्री है — उसका नाम, उसकी GRUB id (`Boot0002` वग़ैरह), और वह सटीक `.efi` फ़ाइल जिस पर वह इशारा करती है। हर लाइन में दो बटन:

बटन	क्या करता है
DEFAULT	इस एंट्री को फ़र्मवेयर की स्थायी डिफ़ॉल्ट बना देता है — अब से हर रीबूट पर यह शुरू होती है।
BOOT NEXT	एक बार की छूट — यह एंट्री सिर्फ़ **अगले** रीबूट पर शुरू होती है, फिर सामान्य क्रम अपने आप फिर से शुरू हो जाता है। जब तक कोई छूट लंबित है, ऊपर एक बैनर दिखता है, जिसमें CANCEL बटन होता है।

WHAT	Boot → EFI टैब → वह एंट्री खोजें जो आप चाहते हैं (जैसे “Odyssey” या “Windows Boot Manager”) → DEFAULT पर क्लिक करें, या एक बार की छूट के लिए BOOT NEXT पर।
WHY	अगर आपके मदरबोर्ड का अपना बूट मेन्यू पूरी तरह ग़लत बूटलोडर शुरू कर रहा है — यह GRUB की सेटिंग नहीं है, फ़र्मवेयर की सेटिंग है — यही एकमात्र जगह है जो इसे ठीक करती है।
RESULT	DEFAULT: एंट्री स्थायी रूप से फ़र्मवेयर के बूट क्रम में सबसे आगे चली जाती है। BOOT NEXT: यह एक बार शुरू होती है, फिर वापस पुरानी स्थिति में आ जाती है।
RISK	कम, और हमेशा इसी टैब से या आपके मदरबोर्ड की अपनी UEFI सेटअप स्क्रीन से (आमतौर पर पावर ऑन पर Del या F2) फिर से ठीक किया जा सकता है, अगर कुछ अनपेक्षित रूप से शुरू हो जाए।

2.4 कर्नेल पैरामीटर

GRUB टैब का आख़िरी सेक्शन प्लेन-टेक्स्ट कर्नेल कमांड लाइन है — जिसे एक सामान्य टेक्स्ट फ़ील्ड की तरह संपादित किया जा सकता है, किसी GRUB सिंटैक्स की ज़रूरत नहीं — और सेकंड में मेन्यू टाइमआउट। **SAVE** इन्हें `/etc/default/grub` में लिखता है; **GRUB-MKCONFIG** इस फ़ाइल से असली बूट मेन्यू को फिर से बनाता है, यही वह कदम है जो सेव किए गए पैरामीटर को अगले बूट पर प्रभावी बनाता है। पैनल फिर से बनाते समय हमेशा आपकी मौजूदा डिफ़ॉल्ट पसंद को दोबारा लागू करता है, इसलिए यह कभी भी यह रीसेट नहीं करता कि डिफ़ॉल्ट रूप से कौन-सा सिस्टम बूट होता है।

```
▸ ADVANCED

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

2.5 Advanced

असल में क्या होता है

Odyssey की GRUB कॉन्फ़िगरेशन **फ़्लैट** है — कोई `submenu` ब्लॉक नहीं। हर कर्नेल एक टॉप-लेवल `menuentry` है, जिसका मतलब है कि एक स्थिर id (कोई संख्यात्मक स्थान नहीं) हमेशा उसे सही तरीक़े से चुन सकता है, कर्नेल अपडेट के बीच भी। सिलेक्टर में जो “Advanced options” समूहीकरण आप देखते हैं वह सिर्फ़ दृश्य सुविधा है, जो किसी ऑपरेटिंग सिस्टम की सरल एंट्री और उसके कर्नेल-वार वैरिएंट के बीच साझा UUID को मिलाकर बनाई गई है — GRUB की अपनी फ़ाइल में यह समूहीकरण नहीं होता।

डिफ़ॉल्ट सेट करना एक ही क्रिया में दो जगह लिखता है: `/etc/default/grub` में स्थायी `GRUB_DEFAULT`, और — क्योंकि `grub.cfg` तुरंत फिर से नहीं बन सकता — ख़ुद `/boot/grub/grub.cfg` में सक्रिय लाइन `set default="..."` का सीधा और सटीक `sed` संपादन, जो `grub-reboot` के इस्तेमाल किए जाने वाले one-shot ब्रांच `${next_entry}` को छोड़ देता है। यही वजह है कि डिफ़ॉल्ट पहले पूरा `grub-mkconfig` किए बिना अगले बूट पर असर करता है।

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

हाथ से OS-PROBER

```
$ sudo xbps-install -y os-prober
# स्कैन चालू करता है — /etc/default/grub में GRUB_DISABLE_OS_PROBER संपादित करता है
$ 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
```

हाथ से EFI — EFIBOOTMGR

EFI टैब `efibootmgr` के लिए एक इंटरफ़ेस है। यह जो कुछ भी करता है, वह एक कमांड से मेल खाता है:

```
# मौजूदा बूट क्रम में सभी फ़र्मवेयर एंट्रीज़ की सूची बनाता है
$ sudo efibootmgr -v
# क्रम बदलता है: एंट्री 0002 को पहले रखता है (= DEFAULT)
$ sudo efibootmgr -o 0002,0000,0001,0003
# एंट्री 0003 पर एक बार का बूट (= BOOT NEXT)
$ sudo efibootmgr -n 0003
# लंबित BootNext रद्द करता है
$ sudo efibootmgr -N
```

अनाथ लोडर

कभी-कभी कोई इंस्टॉलर EFI System Partition पर अपना बूटलोडर लिखता है बिना उसके लिए फ़र्मवेयर एंट्री रजिस्टर किए — लोडर डिस्क पर मौजूद होता है लेकिन फ़र्मवेयर को पता नहीं होता कि वह वहाँ है। Odyssey का पैनल सीधे ESP को स्कैन करता है और उन्हें अलग से **"ON DISK, NOT IN FIRMWARE"** के रूप में सूचीबद्ध करता है, हर एक के साथ एक-क्लिक REGISTER। हाथ से, इसके बराबर है `efibootmgr -c -d /dev/sdX -p N -L "नाम" -l '\EFI\नाम\bootx64.efi'` — ध्यान दें कि नई बनाई गई एंट्रीज़ ख़ुद फ़र्मवेयर द्वारा बूट क्रम में **पहले** रखी जाती हैं, इसलिए अगर आप नहीं चाहते कि यह नियंत्रण ले ले, तो बाद में अपनी सामान्य एंट्री पर फिर से DEFAULT लागू करें।

इस पैनल द्वारा छुई जाने वाली फ़ाइलें

फ़ाइल	भूमिका
`/etc/default/grub`	कर्नेल कमांड लाइन, टाइमआउट, GRUB_DEFAULT, GRUB_DISABLE_OS_PROBER
`/boot/grub/grub.cfg`	असली जनरेट किया गया बूट मेन्यू ख़ुद — इसे कभी हाथ से संपादित न करें; यह grub-mkconfig द्वारा ओवरराइट हो जाता है
EFI वेरिएबल (NVRAM)	फ़र्मवेयर की बूट एंट्रीज़ — यह बिल्कुल भी फ़ाइल नहीं है; पूरी तरह efibootmgr के ज़रिए पढ़ा और लिखा जाता है

CHAPTER 3

Kernel

चार टैब — **SYSCTL**, **CPU GOVERNOR**, **HUGEPAGES**, **MODULES** — साथ ही चारों के ऊपर एक-क्लिक प्रीसेट की एक कतार, जो चारों पर एक साथ लागू होती है।

3.1 दृश्य गाइड — प्रीसेट

ऊपर: आपके चालू कर्नेल का वर्ज़न, सक्रिय शेड्यूलर, और चार प्रीसेट बटन — **DESKTOP**, **GAMING**, **SERVER**, **BALANCED**।

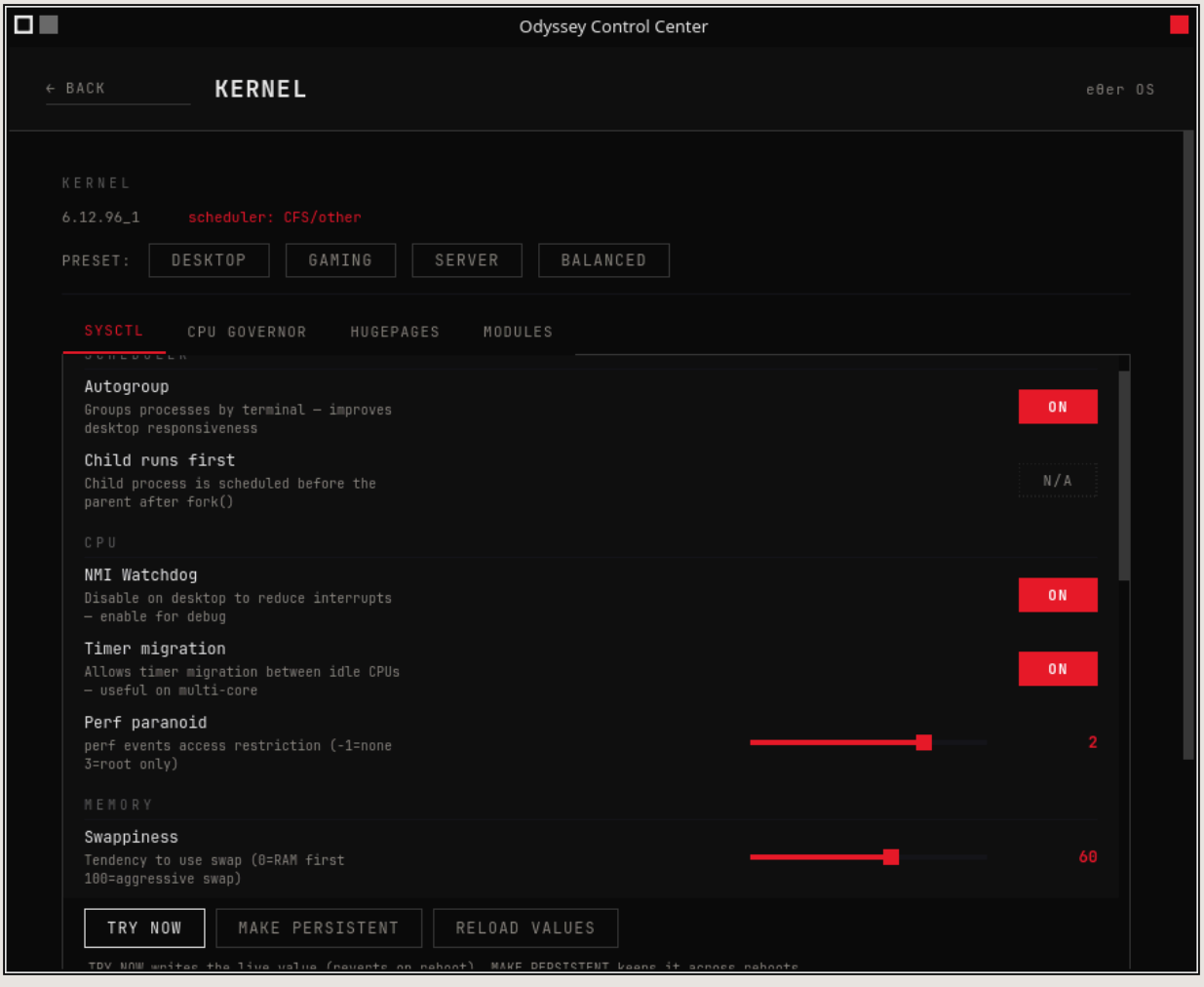

FIG · अभी-अभी खुला SYSCTL टैब: कर्नेल 6.12.96_1, शेड्यूलर CFS/other, चार प्रीसेट, और नीचे Scheduler/CPU/Memory समूह।

प्रीसेट	कब इस्तेमाल करना फ़ायदेमंद है
DESKTOP	रोज़मर्रा की मशीन — ब्राउज़िंग, दफ़्तर का काम, हल्का मल्टीटास्किंग। कम swappiness, मध्यम कैश दबाव, आराम में स्थिर बिजली खपत के लिए टाइमर माइग्रेशन चालू। ज़्यादातर लोगों के लिए समझदारी भरा डिफ़ॉल्ट।

प्रीसेट	कब इस्तेमाल करना फ़ायदेमंद है
GAMING	swappiness लगभग शून्य तक धकेली गई (सब कुछ RAM में रखें), टाइमर माइग्रेशन बंद (कोर काम को माइग्रेट करने के बजाय समर्पित रहते हैं), MangoHud जैसे प्रोफ़ाइलिंग टूल के लिए perf इवेंट पर पाबंदियाँ ढीली की गई हैं। तब इस्तेमाल करें जब फ़्रेम टाइम आराम की बिजली खपत से ज़्यादा मायने रखता हो।
SERVER	ज़्यादा swappiness और कैश दबाव के लिए ज़्यादा सहनशीलता, NMI watchdog चालू, कर्नेल पॉइंटर/लॉग पर सख़्त पाबंदियाँ। ऐसी मशीन पर इस्तेमाल करें जिसमें डिस्प्ले न हो और जो इंटरैक्टिव प्रतिक्रिया की तुलना में स्थिरता और सुरक्षा को प्राथमिकता देती हो।
BALANCED	Desktop और Gaming के बीच का बीच का रास्ता — ऐसी मशीन के लिए जो कभी वर्कस्टेशन होती है, कभी गेमिंग सेशन, और आप बार-बार प्रीसेट बदलना नहीं चाहते।

WHAT	Kernel → DESKTOP, GAMING, SERVER या BALANCED पर क्लिक करें।
WHY	उस व्यक्ति से एक दर्जन sysctl मान हाथ से समायोजित करवाना बहुत ज़्यादा माँग है जो सिर्फ़ "गेमिंग के लिए अच्छी सेटिंग्स" चाहता है — प्रीसेट बिल्कुल यही है, पहले से तय किया हुआ।
RESULT	SYSCTL टैब में हर मान प्रीसेट के नंबरों पर चला जाता है, तैयार और दिखाई देता है — इन्हें सच में लागू करने के लिए आपको अभी भी TRY NOW या MAKE PERSISTENT (नीचे) चाहिए।
RISK	कोई नहीं — प्रीसेट चुनना सिर्फ़ फ़ॉर्म भरता है।

3.2 दृश्य गाइड — SYSCTL, समूह-दर-समूह

नीचे स्क्रॉल करें और हर पैरामीटर एक हेडिंग के नीचे समूहबद्ध है, हर एक के साथ पैनल में सीधे सरल भाषा में विवरण दिया गया है — आपको `vm.compaction_proactiveness` का मतलब समझने के लिए कभी वहाँ पहले पढ़े बिना अंदाज़ा नहीं लगाना पड़ता।

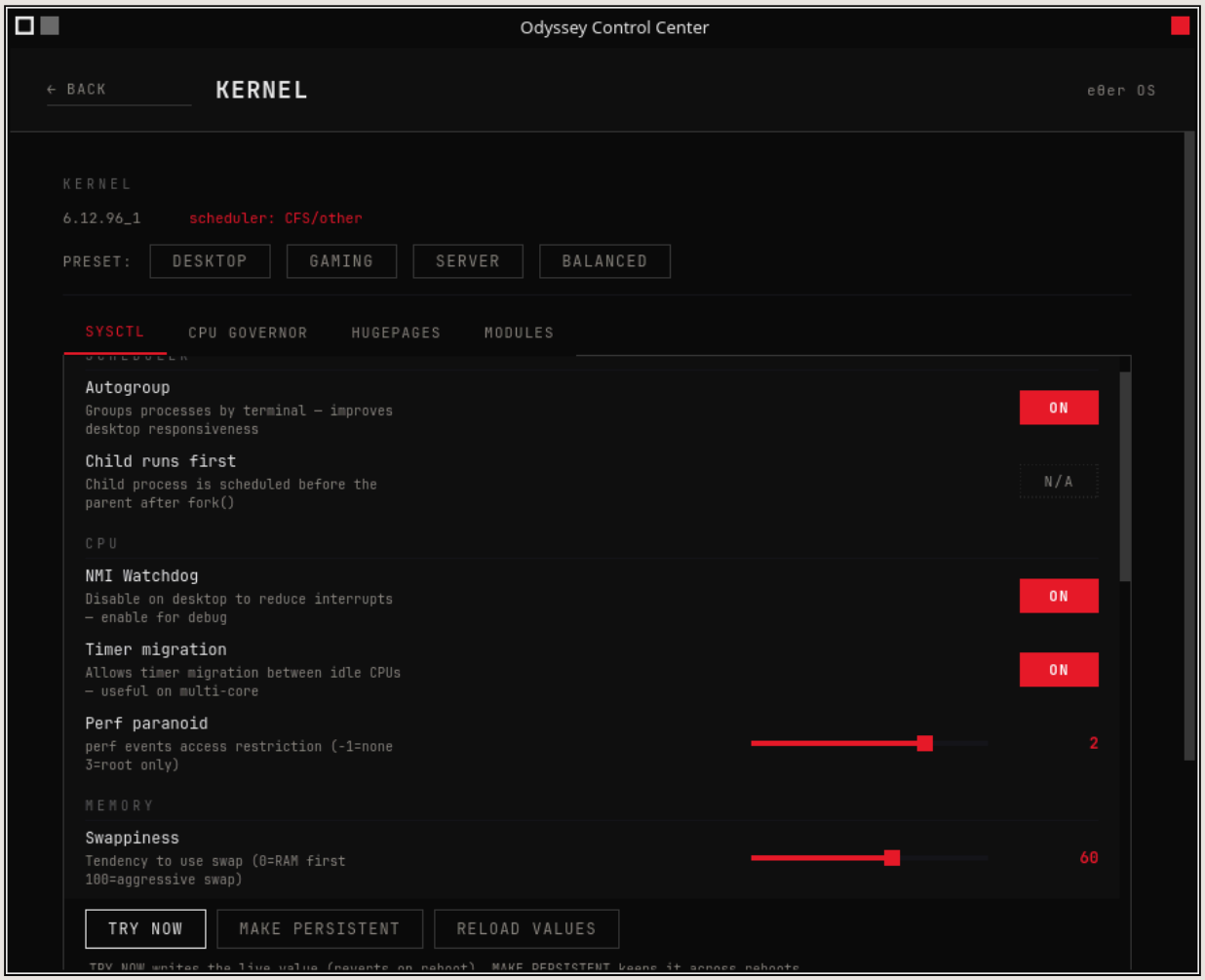

FIG · Scheduler और CPU समूह: Autogroup, Child runs first, NMI Watchdog, Timer migration, Perf paranoid — हर एक के साथ उसका अपना स्विच या स्लाइडर और समझाने वाली एक लाइन।

समूह	क्या शामिल है
Scheduler	प्रोसेस को कैसे समूहबद्ध और प्राथमिकता दी जाती है
CPU	Watchdog, टाइमर माइग्रेशन, परफ़ॉर्मेंस मॉनिटरिंग पर पाबंदियाँ
Memory	Swappiness, कैश दबाव, “गंदे” पेज लिखने की सीमाएँ
Network	TCP Fast Open, backlog क्यू का आकार, सॉकेट बफ़र सीमाएँ
Security	क्या कर्नेल पॉइंटर और लॉग संदेश बिना विशेषाधिकार वाले उपयोगकर्ताओं से छिपाए जाते हैं

नीचे के दो बटन आपके बदलाव लागू करते हैं — और वे जानबूझकर अलग-अलग हैं:

WHAT	TRY NOW
WHY	किसी मान को शून्य जोखिम के साथ आज़माना, कि वह किसी ख़राब रीबूट के बाद बना रहे।
RESULT	`sysctl` के ज़रिए लाइव लिखा जाता है — तुरंत सक्रिय, सिर्फ़ इस सत्र के लिए।
RISK	कोई नहीं — रीबूट पर अपने आप वापस पुरानी स्थिति में आ जाता है।

WHAT	MAKE PERSISTENT
WHY	आपने किसी मान को (या पूरे प्रीसेट को) आज़मा लिया है और चाहते हैं कि वह रीबूट के बाद भी बना रहे।
RESULT	`/etc/sysctl.d/99-odyssey.conf` में लिखा जाता है **और** उसी क्लिक में लाइव लागू भी हो जाता है।
RISK	कम — यह बस एक टेक्स्ट फ़ाइल है; उलटने के लिए लाइन मिटाएँ या फिर से लागू करें।

RELOAD VALUES अभी जो सक्रिय है उसे फिर से पढ़ता है, जो कुछ भी टाइप किया गया लेकिन लागू नहीं हुआ उसे छोड़ देता है।

3.3 दृश्य गाइड — CPU governor

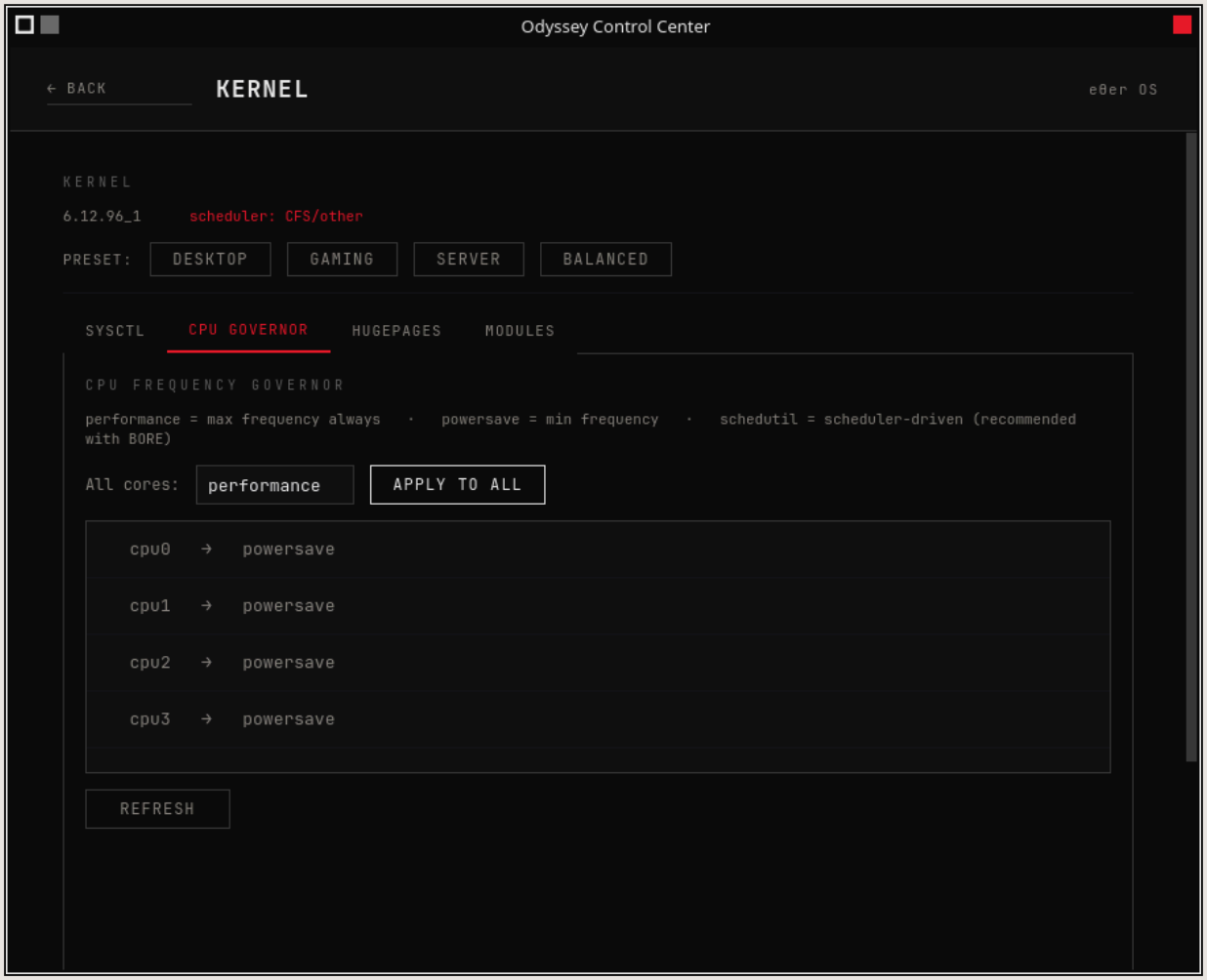

FIG · CPU GOVERNOR टैब: “All cores” को performance पर सेट किया गया है APPLY TO ALL के साथ, और नीचे कोर-वार सूची अभी भी सब कुछ powersave दिखा रही है — एक विसंगति जो सुधार का इंतज़ार कर रही है।

Governor तय करता है कि आपका CPU क्लॉक फ़्रीक्वेंसी को कितनी आक्रामकता से बदलता है। एक ड्रॉपडाउन एक साथ सभी कोर पर governor लागू करता है; नीचे की सूची हर अलग कोर के लिए **लाइव** governor दिखाती है — इसी तरह आप किसी ऐसे कोर को पकड़ते हैं जो ग़लत सेटिंग पर फंसा है, बिल्कुल ऊपर के स्क्रीनशॉट की तरह जहाँ “All cores” performance कहता है लेकिन नीचे हर कोर अभी भी powersave दिखाता है, जिसका मतलब है कि APPLY TO ALL अभी क्लिक नहीं किया गया है।

GOVERNOR	व्यवहार
performance	अधिकतम फ़्रीक्वेंसी, हमेशा — सबसे प्रतिक्रियाशील, सबसे ज़्यादा बिजली खपत
powersave	न्यूनतम फ़्रीक्वेंसी, हमेशा — सबसे शांत, सबसे कम प्रतिक्रियाशील

GOVERNOR	व्यवहार
schedutil	शेड्यूलर के अपने लोड के आकलन से नियंत्रित फ़्रीक्वेंसी — BORE शेड्यूलर के साथ सुझाया गया

ड्रॉपडाउन में सिर्फ़ वही governor दिखते हैं जो आपका ख़ास CPU और कर्नेल असल में उपलब्ध कराते हैं।

3.4 दृश्य गाइड — hugepages

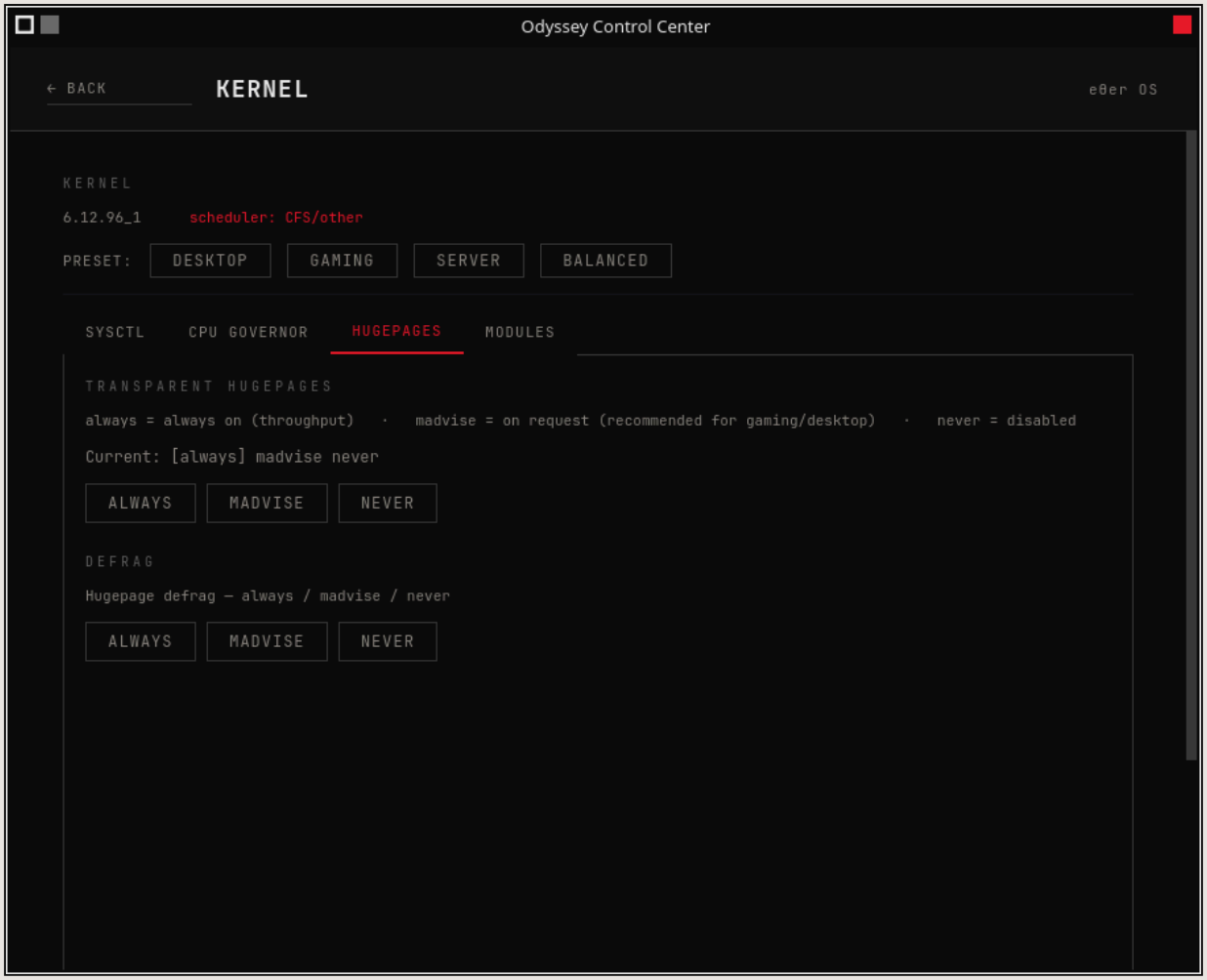

FIG · HUGEPAGES टैब: Transparent Hugepages “Current: [always] madvise never” दिखा रहा है ALWAYS उभरा हुआ, और नीचे Defrag के लिए फिर से वही तीन बटन की कतार।

दो स्वतंत्र सेटिंग्स, हर एक तीन बटन की एक कतार जो क्लिक करते ही लागू हो जाती है — मौजूदा लाइव मान हमेशा बटन के ऊपर वर्ग कोष्ठक में दिखाया जाता है, जैसे स्क्रीनशॉट में `[always] madvise never`।

सेटिंग	क्या नियंत्रित करती है
Transparent hugepages	ALWAYS = हर जगह हमेशा चालू (अधिकतम throughput); MADVISE = सिर्फ़ जब कोई ऐप स्पष्ट रूप से माँगे (गेमिंग/डेस्कटॉप के लिए सुझाया गया); NEVER = पूरी तरह बंद
Defrag	माँगे जाने पर hugepages को **बनाने** के लिए कर्नेल कितनी आक्रामकता से मेमोरी को संकुचित करता है — वही तीन विकल्प, ऊपर की सेटिंग से स्वतंत्र

3.5 दृश्य गाइड — modules

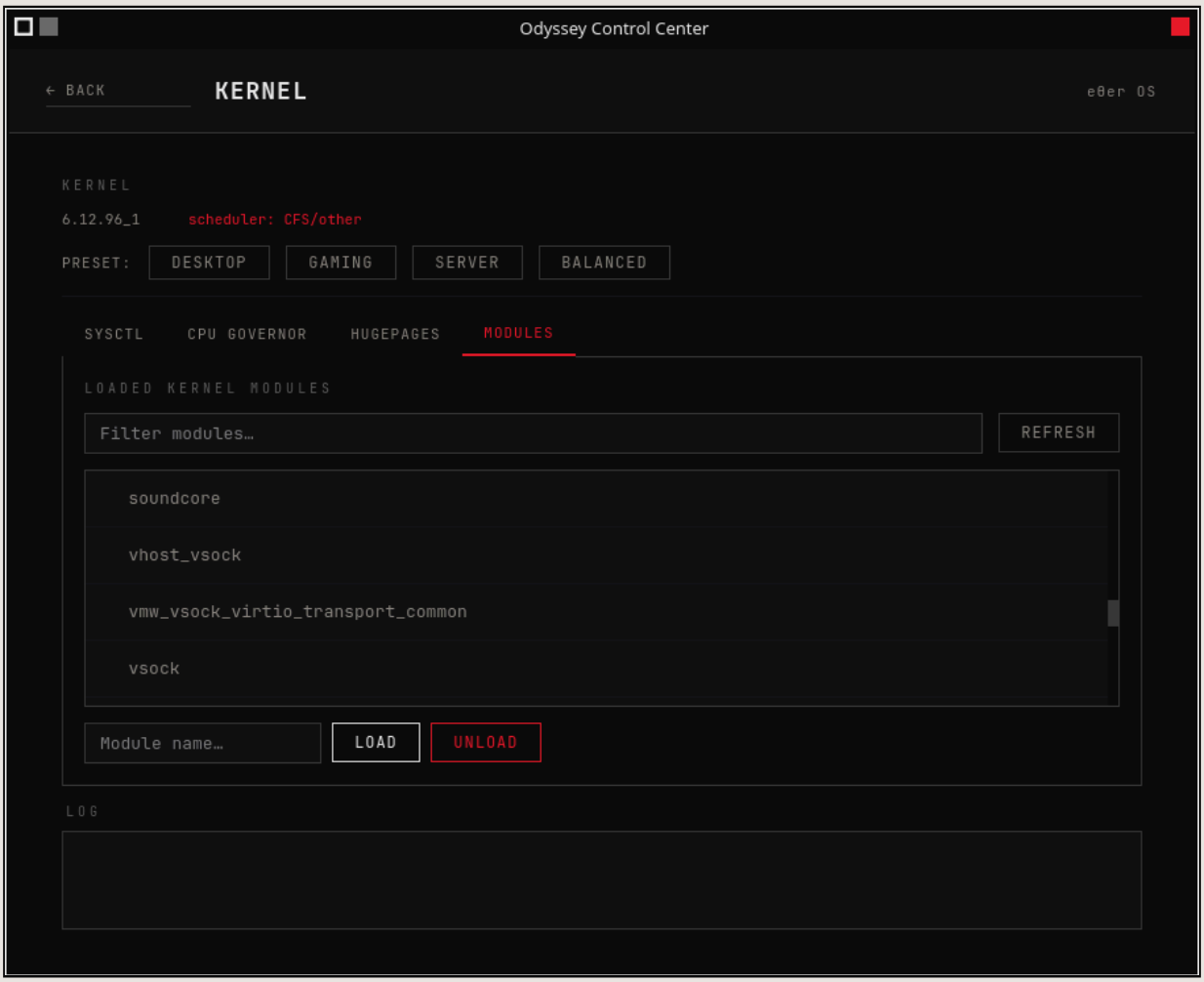

FIG · MODULES टैब: एक फ़िल्टर फ़ील्ड, एक लाइव सूची (soundcore, vhost_vsock, vsock…), और नीचे नाम से टाइप किए गए मॉड्यूल के लिए LOAD/UNLOAD की एक कतार।

अभी लोड किए गए हर कर्नेल मॉड्यूल की एक लाइव, फ़िल्टर करने योग्य सूची — इसे सीमित करने के लिए फ़िल्टर में टाइप करें, फिर से पढ़ने के लिए REFRESH। बाकी तीन टैब के विपरीत, यह आपको सीधे कार्रवाई करने देता है: एक मॉड्यूल का नाम टाइप करें और **LOAD** या **UNLOAD** पर क्लिक करें।

WHAT	Kernel → MODULES टैब → एक मॉड्यूल का नाम टाइप करें → LOAD या UNLOAD।
WHY	कभी-कभी आपको किसी ख़ास ड्राइवर या फ़ंक्शन मॉड्यूल को बिना रीबूट किए सक्रिय (या हटाया हुआ) चाहिए होता है — हार्डवेयर की जाँच करना, किसी डिवाइस को छुड़ाना जिसे कोई दूसरा मॉड्यूल पकड़े हुए है।
RESULT	मॉड्यूल तुरंत चालू कर्नेल में डाला जाता है या उसमें से हटा दिया जाता है।
RISK	ख़ासतौर पर UNLOAD के लिए मध्यम — किसी ऐसे मॉड्यूल को हटाना जिस पर कुछ और निर्भर हो, वह अस्थिर कर सकता है जो उसका इस्तेमाल कर रहा था (कोई डिस्प्ले ड्राइवर, कोई फ़ाइल सिस्टम)। LOAD आम तौर पर सुरक्षित है; कर्नेल किसी असंगत मॉड्यूल को आधा लागू करने के बजाय उसे लोड करने से सीधे इनकार कर देता है।

▸ **ADVANCED**

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

3.6 Advanced

सटीक SYSCTL कीज़

की	पैनल में लेबल
`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

प्रीसेट मान, DESKTOP बनाम 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
# पूरी प्रीसेट टेबल सीधे panel_kernel.py में पढ़ी जा सकती हैं —
# अगर आप हर फ़ील्ड चाहते हैं तो Control Center का सोर्स कोड खोलें
```

हाथ से, पैनल के बिना

```
# लाइव, अस्थायी
$ sudo sysctl vm.swappiness=10
# स्थायी — वही फ़ाइल जो पैनल लिखता है
$ echo “vm.swappiness=10” | sudo tee -a /etc/sysctl.d/99-odyssey.conf
$ sudo sysctl -system
# CPU governor, कोर-वार
$ 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
# मॉड्यूल
$ sudo modprobe MODULE_NAME
$ sudo modprobe -r MODULE_NAME
```

TRY NOW और MAKE PERSISTENT अलग-अलग क्रियाएँ क्यों हैं

`sysctl` की लिखाई अपने स्वभाव से हमेशा सिर्फ़ लाइव होती है — कर्नेल में कोई अंतर्निहित स्थायित्व नहीं है। MAKE PERSISTENT Odyssey का एक ही मानक फ़ाइल चुनना है, `/etc/sysctl.d/99-odyssey.conf`, और लाइव लिखाई तथा फ़ाइल लिखाई दोनों को एक साथ करना ताकि दोनों कभी अलग न हो जाएँ। इस फ़ाइल को सीधे संपादित करना और बाद में `sysctl --system` चलाना, बटन पर क्लिक करने जैसा ही असर रखता है।

CHAPTER 4

Memory & Swap

एक ही स्क्रीन, तीन सेक्शन: केवल-पढ़ने वाला सारांश, swappiness, और ZRAM।

4.1 दृश्य गाइड

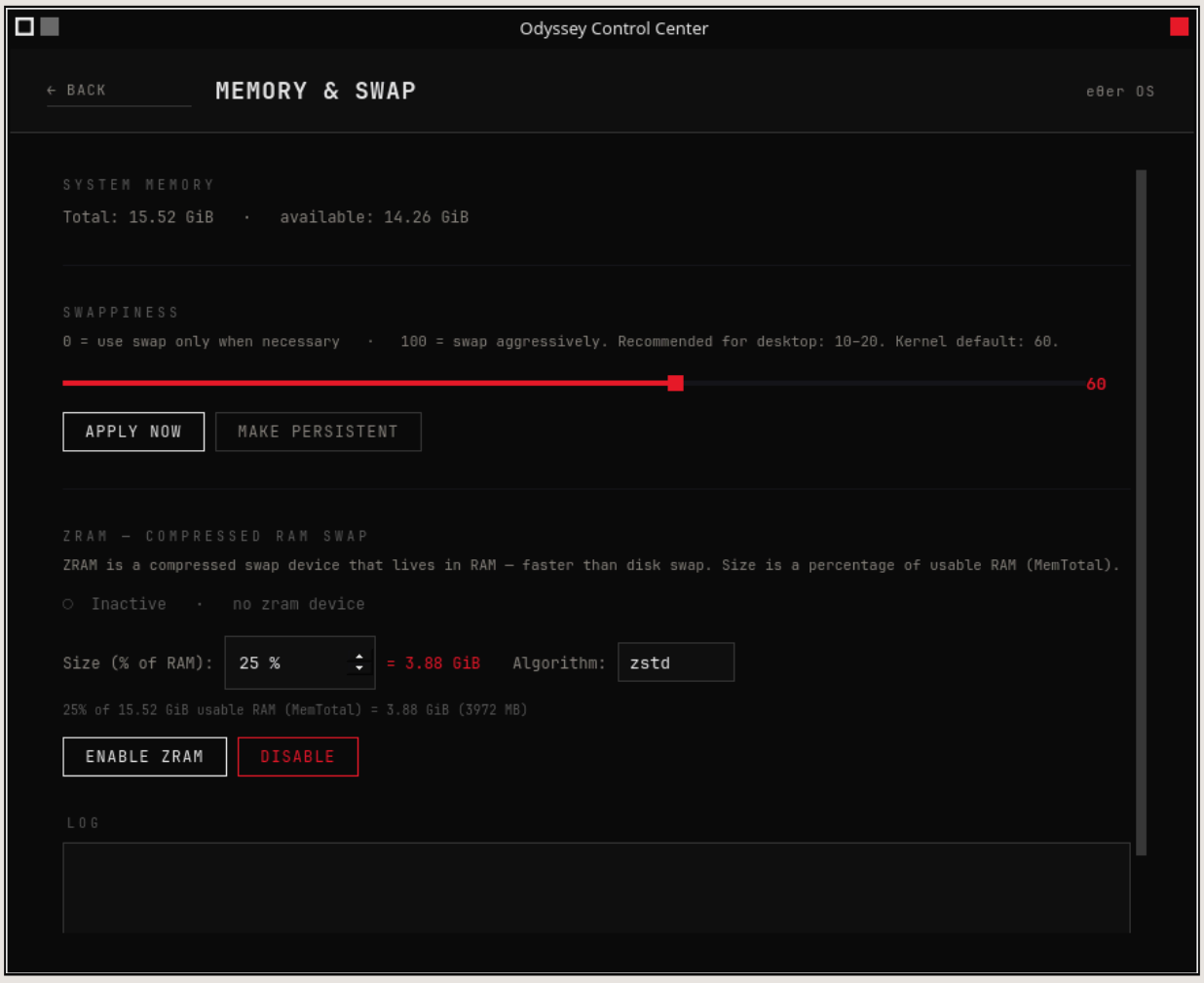

FIG · पूरा पैनल: System memory (कुल 15.52 GiB, उपलब्ध 14.26 GiB), Swappiness स्लाइडर 60 पर APPLY NOW / MAKE PERSISTENT के साथ, और नीचे ZRAM “Inactive · no zram device” दिखाता हुआ, आकार, एल्गोरिद्म और ENABLE/DISABLE के साथ।

SYSTEM MEMORY

ऊपर सिर्फ़ एक सारांश — कुल RAM और अभी क्या उपलब्ध है। कोई नियंत्रण नहीं; यह नीचे के दो सेक्शन के लिए संदर्भ है।

SWAPPINESS

0 से 100 तक का एक ही स्लाइडर। 0 का मतलब है “swap का इस्तेमाल तभी करें जब सच में कोई और चारा न हो”; 100 का मतलब है “RAM भरने से बहुत पहले ही आक्रामक तरीक़े से swap का इस्तेमाल करें”। कर्नेल का डिफ़ॉल्ट 60 है (जो स्क्रीनशॉट दिखाता है); पैनल का अपना सुझाव डेस्कटॉप के लिए 10–20 बताता है — वही संख्या जो Kernel पैनल का DESKTOP प्रीसेट सेट करता है।

WHAT	स्लाइडर को खींचें → आज़माने के लिए APPLY NOW, संतुष्ट होने पर MAKE PERSISTENT।
WHY	कम मान वह चीज़ जिसे आप सक्रिय रूप से इस्तेमाल कर रहे हैं उसे धीमे swap में ले जाने के बजाय तेज़ RAM में रखता है — काफ़ी मेमोरी इंस्टॉल होने पर आमतौर पर यही चाहिए होता है।
RESULT	APPLY NOW इसे लाइव बदलता है, तुरंत, सिर्फ़ इस सत्र के लिए। MAKE PERSISTENT इसे एक कॉन्फ़िगरेशन फ़ाइल में लिखता है ताकि यह रीबूट के बाद भी बना रहे।
RISK	बहुत कम। सच में RAM की कमी वाली मशीन पर एक बेहद कम मान उसे भारी लोड में धीरे-धीरे swap करने के बजाय अचानक मेमोरी ख़त्म होने के लिए ज़्यादा प्रवण बना सकता है — अगर ऐसा हो तो इसे फिर से ऊपर कर दें।

ZRAM — RAM में संकुचित स्वैप

ZRAM एक ऐसा स्वैप डिवाइस है जो RAM के **अंदर** ही रहता है, संकुचित रूप में — डिस्क पर swap से काफ़ी तेज़ क्योंकि कोई डिस्क शामिल ही नहीं होती। आकार आपकी कुल उपलब्ध RAM का एक प्रतिशत है, कोई तय संख्या नहीं, इसलिए यह सेटिंग अलग-अलग मशीनों पर समझदारी से पैमाने पर फ़िट होती है; आकार फ़ील्ड के बगल में एक लाइव रीडिंग उस प्रतिशत को आपके लिए GiB में बदल देती है (स्क्रीनशॉट में: 25% = 15.52 GiB वाले सिस्टम पर 3.88 GiB)।

WHAT	आकार का प्रतिशत सेट करें और एक एल्गोरिद्म चुनें (zstd, lz4, lzo, या deflate) → ENABLE ZRAM।
WHY	RAM में संकुचित swap मेमोरी के दबाव को डिस्क पर swap से कहीं कम परफ़ॉर्मेंस असर के साथ सोख लेता है — किसी भी मशीन पर उपयोगी, ख़ासकर सीमित RAM वाली मशीन पर।
RESULT	चुने गए आकार और एल्गोरिद्म के साथ एक `zram0` डिवाइस बनाया और सक्रिय किया जाता है; स्टेटस लाइन “Inactive” से सक्रिय में बदल जाती है।
RISK	कम। zstd (डिफ़ॉल्ट) ज़्यादातर मशीनों के लिए संकुचन अनुपात और CPU लागत के बीच अच्छा संतुलन रखता है। DISABLE इसे किसी भी समय साफ़-सुथरे तरीक़े से बंद कर देता है।

▸ ADVANCED

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

4.2 Advanced

यहाँ “हार्डवेयर पर निर्भर करता है” का असल मतलब क्या है

दोनों सेटिंग्स आपके ख़ास हार्डवेयर के साथ इंटरैक्ट करती हैं, सिर्फ़ “ज़्यादा RAM = बेहतर” के साथ नहीं:

- **तेज़ NVMe SSD**: 10–20 की रेंज में swappiness बहुत कम कीमत रखती है, भले ही swap हो — डिस्क इतनी तेज़ है कि यह लगभग महसूस नहीं होता। ZRAM फिर भी लेटेंसी और SSD की टूट-फूट में जीतता है, लेकिन मैकेनिकल डिस्क की तुलना में फ़ासला छोटा होता है।
- **स्वैप के तौर पर मैकेनिकल HDD या eMMC**: swappiness को कम रखें (0–10) — धीमे माध्यमों पर डिस्क swap उस झटकेदार अनुभव का कारण बनता है जिसे “thrashing” कहते हैं। ठीक यहीं ZRAM कहीं ज़्यादा बड़ा फ़ायदा देता है।
- **CPU की गुंजाइश**: ZRAM डिस्क I/O को संकुचन और डी-संकुचन पर ख़र्च होने वाले CPU चक्रों से बदलता है। CPU-सीमित मशीन पर (पुराना हार्डवेयर, या गेमिंग के दौरान पहले से भरा हुआ), `lz4` `zstd` की तुलना में ज़्यादा तेज़ी से संकुचित करता है लेकिन ख़राब अनुपात के साथ — अगर CPU समय ज़्यादा तंग संसाधन है तो यह एक समझदार समझौता है। किसी भी आधुनिक हार्डवेयर पर, `zstd` का बेहतर अनुपात उसकी थोड़ी ज़्यादा लागत के लायक है।

- **RAM की गुंजाइश**: कुल सिर्फ़ 8 GiB वाली मशीन पर एक बड़ा ZRAM प्रतिशत (जैसे 50%) ऐप्लिकेशन के लिए कम असली RAM छोड़ता है, क्योंकि ZRAM ख़ुद संकुचित पेज संग्रहीत करने के लिए असली मेमोरी का इस्तेमाल करता है। 16 GiB या उससे ज़्यादा पर, 25% गुंजाइश के साथ एक आरामदायक डिफ़ॉल्ट है।

हाथ से

```
# swappiness, लाइव
$ sudo sysctl vm.swappiness=15
# swappiness, स्थायी — वही फ़ाइल जो पैनल लिखता है
$ echo “vm.swappiness=15” | sudo tee /etc/sysctl.d/99-odyssey.conf
# zram, हाथ से कॉन्फ़िगरेशन
$ 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
# पुष्टि करें कि यह सक्रिय है
$ swapon -show
```

पैनल ZRAM0 को लाइव आकार बदलने के बजाय रीसेट क्यों करता है

किसी ZRAM डिवाइस का आकार तब तक नहीं बदला जा सकता जब तक वह सक्रिय है और swap के तौर पर इस्तेमाल हो रहा है — कर्नेल को पहले उसे बंद और रीसेट करने की ज़रूरत होती है। पैनल का ENABLE ZRAM बटन इस पूरे क्रम को अपने आप चलाता है (swapoff, reset, फिर से कॉन्फ़िगर, swapon), यही वजह है कि पहले से सक्रिय डिवाइस पर आकार या एल्गोरिद्म बदलना फिर भी एक साफ़-सुथरी क्रिया जैसा दिखता है, ग़लती जैसा नहीं।

CHAPTER 5

Firewall

Firewall पैनल का अपना समर्पित अध्याय है, पूरी गहराई के साथ, **Security and Hardening Guide** में — नियम-सेट को लाइन-दर-लाइन समझाया गया है, सभी 21 प्रीसेट अपने-अपने जोखिमों के साथ, कच्ची कॉन्फ़िगरेशन एडिटर, और कमांड-लाइन के बराबर विकल्प। यह अध्याय दृश्य रूप से मार्गदर्शित संस्करण है।

5.1 दृश्य गाइड

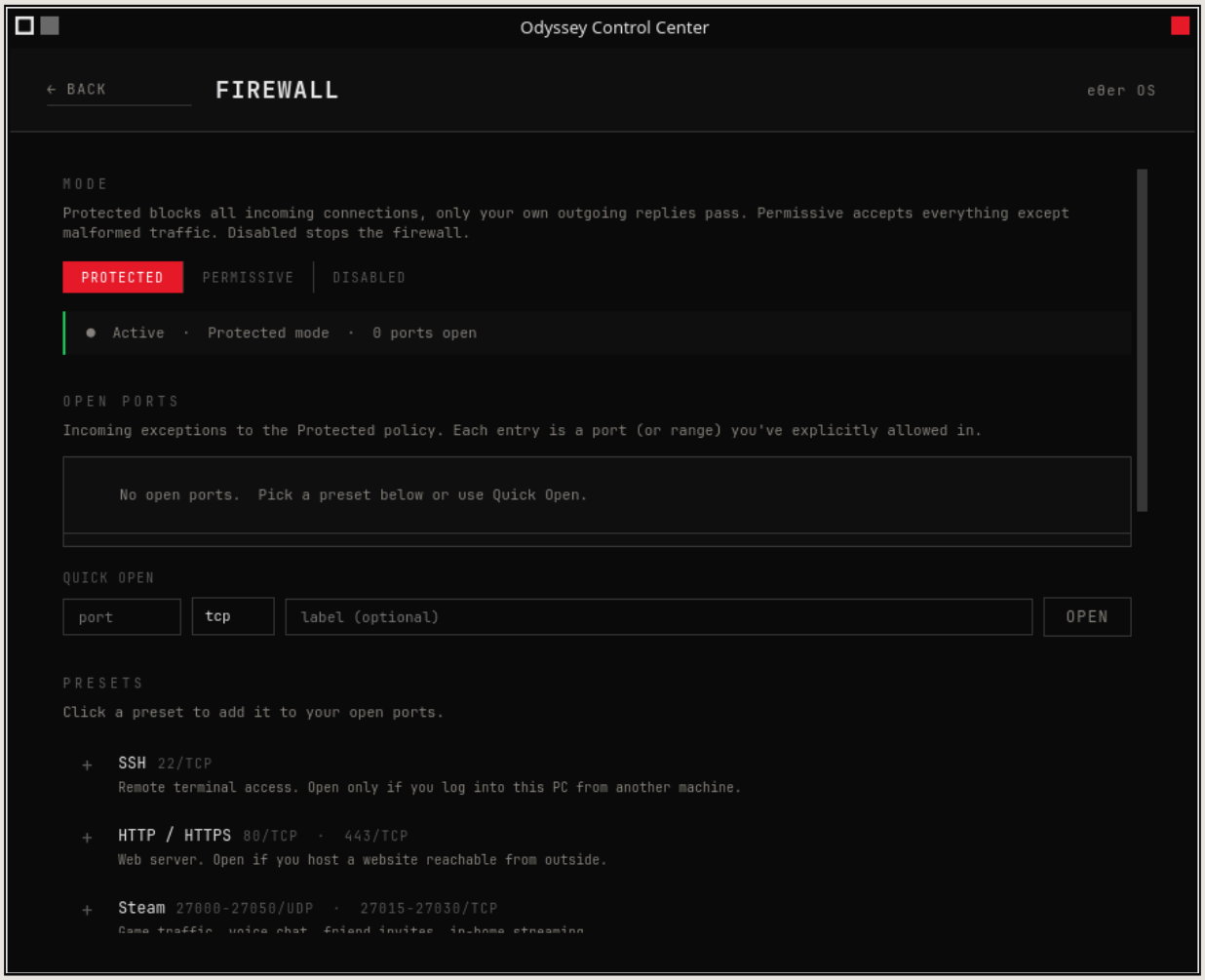

FIG · PROTECTED मोड सक्रिय, “0 ports open”, Quick Open फ़ील्ड्स, और SSH, HTTP/HTTPS और Steam से शुरू होने वाले प्रीसेट टाइल्स।

MODE

ऊपर तीन बटन हैं, और पैनल हर एक को उनके ठीक ऊपर एक लाइन में समझाता है:

मोड	क्या करता है
PROTECTED	डिफ़ॉल्ट। सभी आने वाले कनेक्शन ब्लॉक करता है; सिर्फ़ आपके अपने बाहर जाने वाले जवाब गुज़रते हैं।
PERMISSIVE	ख़राब फ़ॉर्मेट वाले ट्रैफ़िक को छोड़कर सब कुछ स्वीकार करता है — "क्या फ़ायरवॉल इसे तोड़ रहा है?" जाँचने के लिए एक डायग्नोस्टिक सेटिंग, कोई ऐसी जगह नहीं जहाँ रुके रहना चाहिए।
DISABLED	फ़ायरवॉल को पूरी तरह बंद कर देता है।

नीचे की स्टेटस लाइन पुष्टि करती है कि असल में क्या सक्रिय है — स्क्रीनशॉट में, **"Active · Protected mode · 0 ports open"**।

OPEN PORTS और प्रीसेट

OPEN PORTS आपके स्पष्ट अपवादों की सूची बनाता है — डिफ़ॉल्ट रूप से ख़ाली, जैसा दिखाया गया है। **QUICK OPEN** सीधे पोर्ट, प्रोटोकॉल और वैकल्पिक लेबल लेता है, उस हर चीज़ के लिए जो प्रीसेट सूची में नहीं है। नीचे **PRESETS** आठ आम सेवाओं को हर एक क्लिक के साथ पेश करता है, हर एक के साथ एक लाइन जो बताती है कि आपको इसकी असल में कब ज़रूरत होगी (SSH: "रिमोट टर्मिनल एक्सेस, इसे तभी खोलें जब आप किसी दूसरी मशीन से इस PC में लॉगिन करते हों"), साथ ही एक विस्तार योग्य "MORE PRESETS" के नीचे तेरह और।

WHAT	Firewall → किसी प्रीसेट टाइल पर क्लिक करें, या QUICK OPEN भरें → SAVE & APPLY।
WHY	आप कुछ चला रहे हैं — कोई गेम सर्वर, मीडिया सर्वर, डेवलपमेंट सर्वर — जिसे किसी दूसरी मशीन से पहुँचना है।
RESULT	वह पोर्ट खुल जाता है; बाकी सब कुछ PROTECTED नीति के तहत बंद रहता है।
RISK	पूरी तरह प्रीसेट पर निर्भर करता है — इंटरनेट से पहुँच योग्य किसी मशीन पर SSH या HTTP/HTTPS से आगे कुछ भी खोलने से पहले Security and Hardening Guide में हर प्रीसेट के जोखिम नोट्स देखें।

जब तक आप **SAVE & APPLY** नहीं दबाते, कुछ भी लागू नहीं होता — बदलाव पहले लंबित स्थिति में जाते हैं, ताकि पुष्टि करने से पहले आप ठीक वही जाँच सकें जो आप खोलने वाले हैं।

```
▸ ADVANCED

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

5.2 Advanced

पूरा तकनीकी संदर्भ — `nftables.conf` का डिफ़ॉल्ट नियम-सेट नियम- दर-नियम समझाया गया, 21 प्रीसेट में से हर एक अपने ख़ास एक्सपोज़र और कम करने के उपाय के साथ, सिर्फ़ LAN तक पहुँच सीमित करने के नुस्खे, `nft` कमांड-लाइन के बराबर विकल्प, और बाहर लॉक हुए SSH सेशन को ठीक करना — यह सब **Security and Hardening Guide के अध्याय 2** में है (आपके इंस्टॉल किए गए सिस्टम पर `/usr/share/doc/odyssey/`)। यह मैनुअल जानबूझकर इसे दोहराता नहीं है; उस अध्याय को इस सेक्शन की अगली कड़ी मानें।

```
# वह कमांड जिसे अभी जानना फ़ायदेमंद है:
$ sudo nft list ruleset
# बिल्कुल दिखाता है कि कर्नेल में क्या लोड है, पैनल से स्वतंत्र
```

CHAPTER 6

AppArmor

पाँच टैब: **STATUS**, **PROFILES**, **COMMUNITY**, **VIOLATIONS**, **EDITOR**। Firewall जैसा ही क्रॉस-रेफरेंस: अनिवार्य एक्सेस कंट्रोल असल में क्या है, कन्फ़ाइनमेंट का पूरा जीवनचक्र, एक व्यावहारिक उदाहरण — इन सबकी गहरी जानकारी Security and Hardening Guide में है। यहाँ दृश्य गाइड है।

6.1 दृश्य गाइड — STATUS

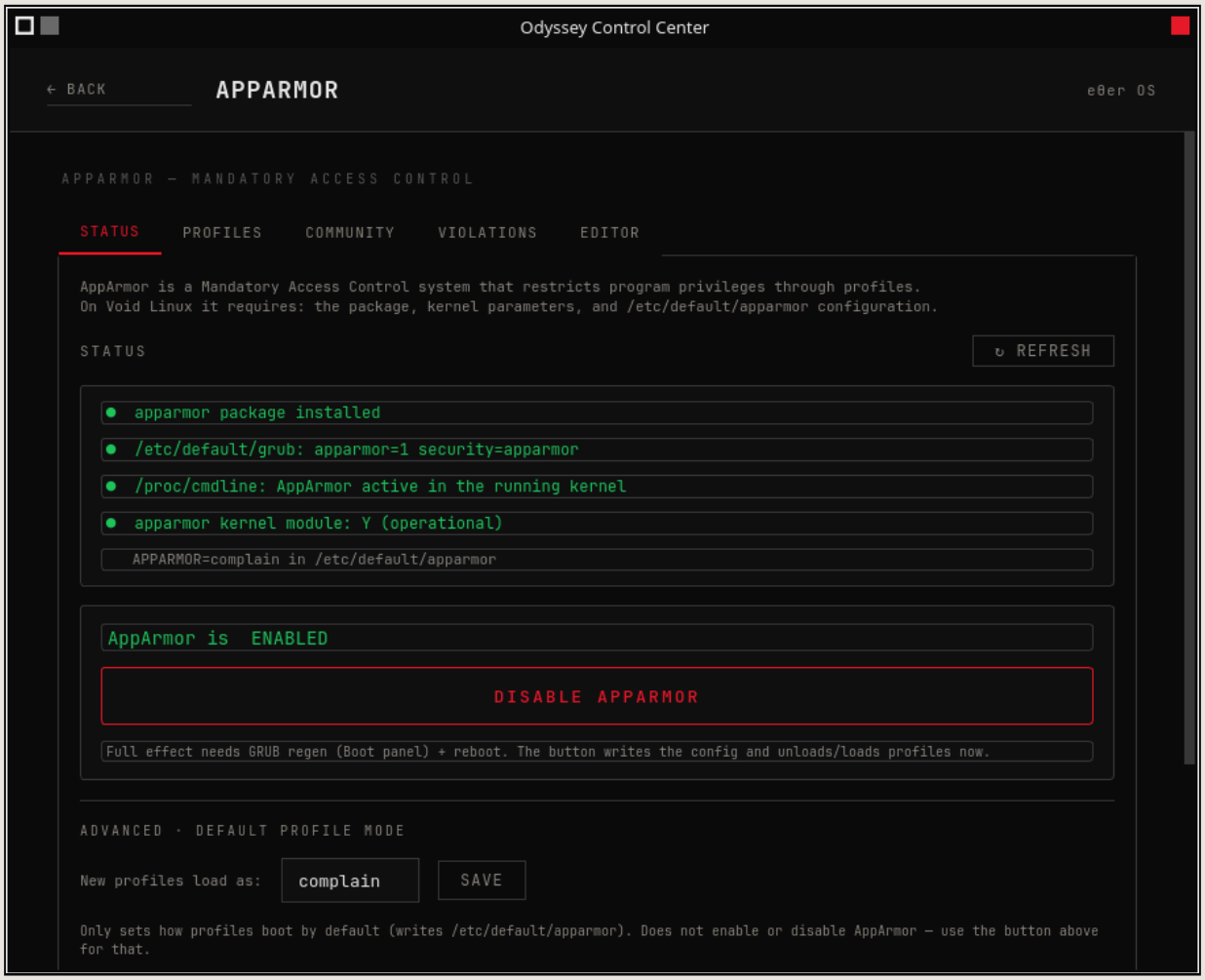

FIG · STATUS टैब: चार हरे चेकमार्क (पैकेज इंस्टॉल, GRUB पैरामीटर, चालू कर्नेल, कर्नेल मॉड्यूल), लाल DISABLE APPARMOR बटन के साथ “AppArmor is ENABLED”, और डिफ़ॉल्ट प्रोफ़ाइल मोड complain पर सेट।

चार जाँचें ऊपर से नीचे तक पुष्टि करती हैं कि यह फ़्रेमवर्क असल में काम कर रहा है: `apparmor` पैकेज इंस्टॉल है, GRUB में कर्नेल पैरामीटर सेट हैं, **चालू** कर्नेल असल में उन्हीं के साथ शुरू हुआ है (यह वह है जो पीछे रह सकती है अगर आपने उन्हें चालू करने के बाद रीबूट नहीं किया), और कर्नेल मॉड्यूल ख़ुद चालू हालत में है। नीचे: एक बड़ा चालू/बंद बटन, और **ADVANCED · DEFAULT PROFILE MODE** — क्या नई लोड हुई प्रोफ़ाइलें complain या enforce में शुरू होती हैं।

WHAT	AppArmor → STATUS → पुष्टि करें कि चारों जाँचें हरी हैं।
WHY	अगर कोई जाँच लाल या पीली है, तो इस पैनल में कुछ और सही तरीक़े से काम नहीं कर सकता — जब AppArmor के साथ कुछ ग़लत लगे तो यही हमेशा सबसे पहले देखने वाली चीज़ है।
RESULT	यह विश्वास कि फ़्रेमवर्क वाक़ई सक्रिय है, सिर्फ़ इंस्टॉल ही नहीं।
RISK	कोई नहीं — केवल-पढ़ने वाला सत्यापन।

6.2 दृश्य गाइड — PROFILES

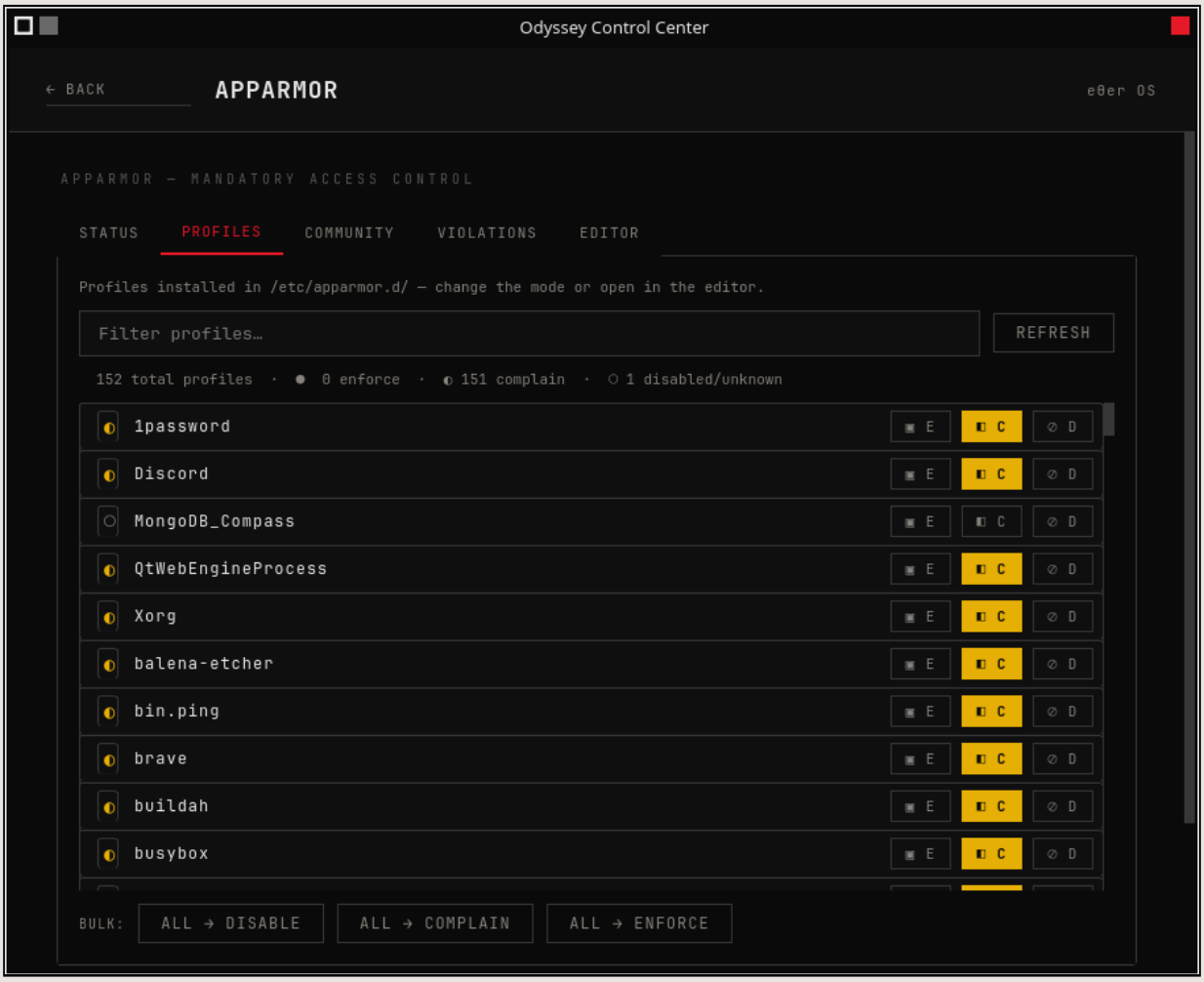

FIG · PROFILES टैब: कुल 152 प्रोफ़ाइलें, 0 enforce, 151 complain, 1 disabled/unknown — 1password, Discord, MongoDB_Compass और दूसरों के साथ एक फ़िल्टर करने योग्य सूची, हर एक के साथ E / C / D बटन, और नीचे ALL→DISABLE / ALL→COMPLAIN / ALL→ENFORCE।

हर लोड की गई प्रोफ़ाइल की एक लाइन, ऊपर एक फ़िल्टर फ़ील्ड और गिनती के साथ (स्क्रीनशॉट में: 152 प्रोफ़ाइलें, एक को छोड़कर सब complain मोड में — Odyssey का डिफ़ॉल्ट)। हर लाइन में तीन बटन:

बटन	असर
E	इस प्रोफ़ाइल को enforce में बदलता है — उल्लंघन ब्लॉक होते हैं, सिर्फ़ दर्ज नहीं होते
C	complain में बदलता है — उल्लंघन दर्ज करता है, कुछ भी ब्लॉक नहीं करता
D	प्रोफ़ाइल को पूरी तरह बंद कर देता है — प्रोग्राम बिना किसी पाबंदी के चलता है

नीचे के तीन **BULK** बटन सूची की हर प्रोफ़ाइल पर एक साथ वही बदलाव लागू करते हैं — COMMUNITY से एक साथ इंस्टॉल करने के तुरंत बाद उपयोगी, लेकिन ALL→ENFORCE को ख़ास सावधानी से बरतें; अगले सेक्शन में ऐक्शन ब्लॉक देखें।

6.3 दृश्य गाइड — COMMUNITY

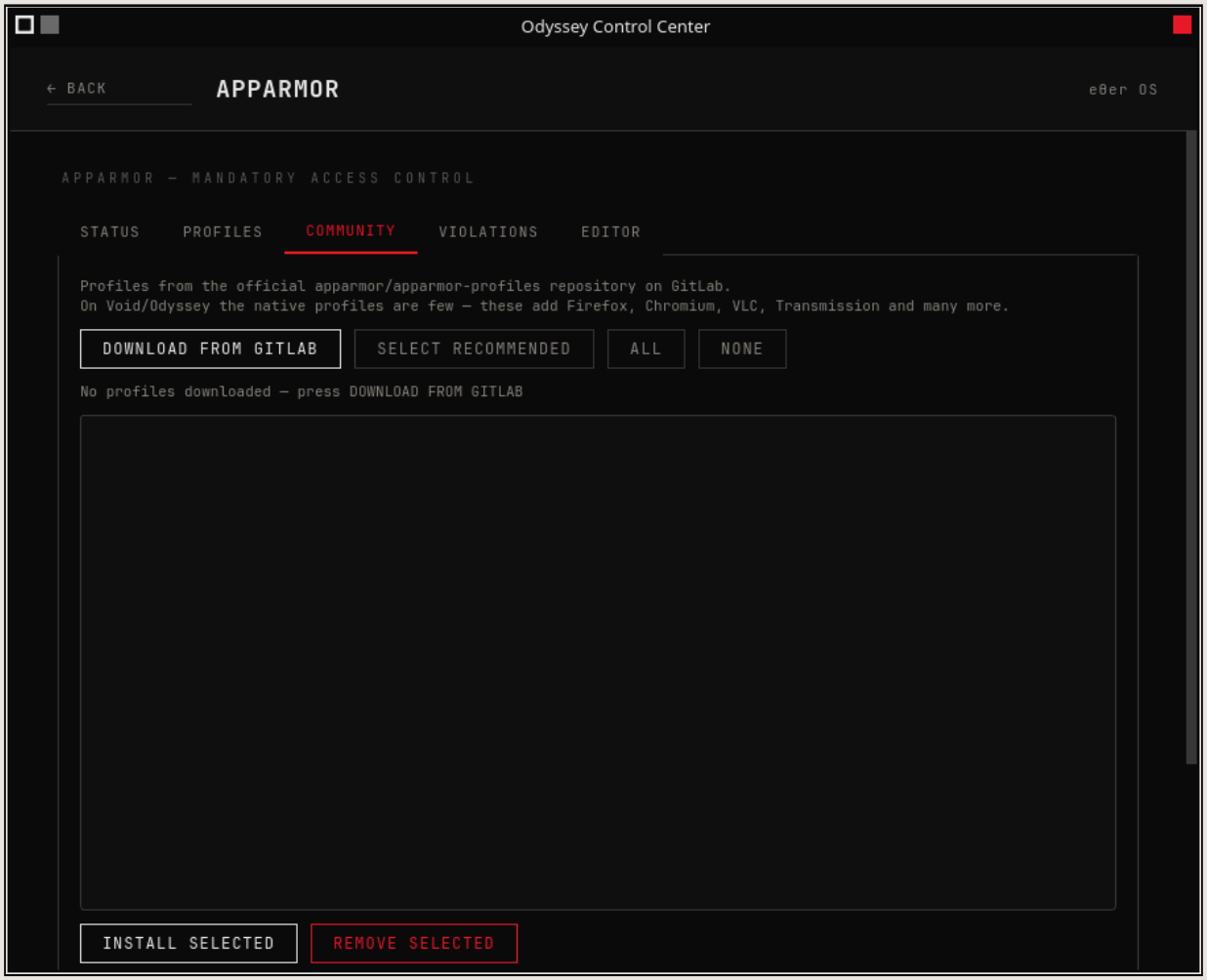

FIG · COMMUNITY टैब: “No profiles downloaded — press DOWNLOAD FROM GITLAB”, DOWNLOAD FROM GITLAB, SELECT RECOMMENDED, ALL और NONE बटन के साथ, और नीचे INSTALL SELECTED / REMOVE SELECTED।

AppArmor डिफ़ॉल्ट रूप से मुख्य रूप से सिस्टम के हिस्सों को सीमित करता है — जिन डेस्कटॉप ऐप्लिकेशन की आपको असल में चिंता है (ब्राउज़र, चैट ऐप्स, मीडिया प्लेयर) उन्हें कम्युनिटी से प्रोफ़ाइलें चाहिए। यह टैब उन्हें सीधे GitLab पर आधिकारिक रिपॉज़िटरी `apparmor/apparmor-profiles` से लाता है।

1. **DOWNLOAD FROM GITLAB**

   उपलब्ध प्रोफ़ाइलों की मौजूदा सूची लाता है — अभी कुछ भी इंस्टॉल नहीं होता, यह बस सूची भरता है।

2. **SELECT RECOMMENDED (या ALL / NONE)**

   डेस्कटॉप के लिए एक समझदार डिफ़ॉल्ट सेट चुनता है — Firefox परिवार के ब्राउज़र, Chromium, VLC, Transmission और इसी तरह के।

3. **INSTALL SELECTED**

   चुनी गई प्रोफ़ाइलों को इंस्टॉल करता है — वे बाकी सब की तरह complain मोड में लोड होती हैं, नीचे वाले “पहले देखो-फिर लागू करो” चक्र के लिए तैयार।

WHAT	AppArmor → COMMUNITY → DOWNLOAD FROM GITLAB → SELECT RECOMMENDED (या अलग-अलग चुनें) → INSTALL SELECTED।
WHY	जो ऐप्लिकेशन अविश्वसनीय इनपुट के सबसे ज़्यादा संपर्क में रहते हैं — कोई ब्राउज़र जो कोई ख़तरनाक पेज रेंडर करता है, कोई मीडिया प्लेयर जो कोई गढ़ी हुई फ़ाइल खोलता है — वही होते हैं जिन्हें एक प्रोफ़ाइल सबसे बेहतर सुरक्षा देती है।
RESULT	नई प्रोफ़ाइलें PROFILES में दिखने लगती हैं, complain मोड में, यह दर्ज करते हुए कि वे क्या ब्लॉक करतीं, बिना अभी कुछ भी ब्लॉक किए।
RISK	इंस्टॉल करते समय कोई नहीं — जब तक आप PROFILES में तय नहीं करते, कुछ भी लागू नहीं होता।

6.4 दृश्य गाइड — VIOLATIONS

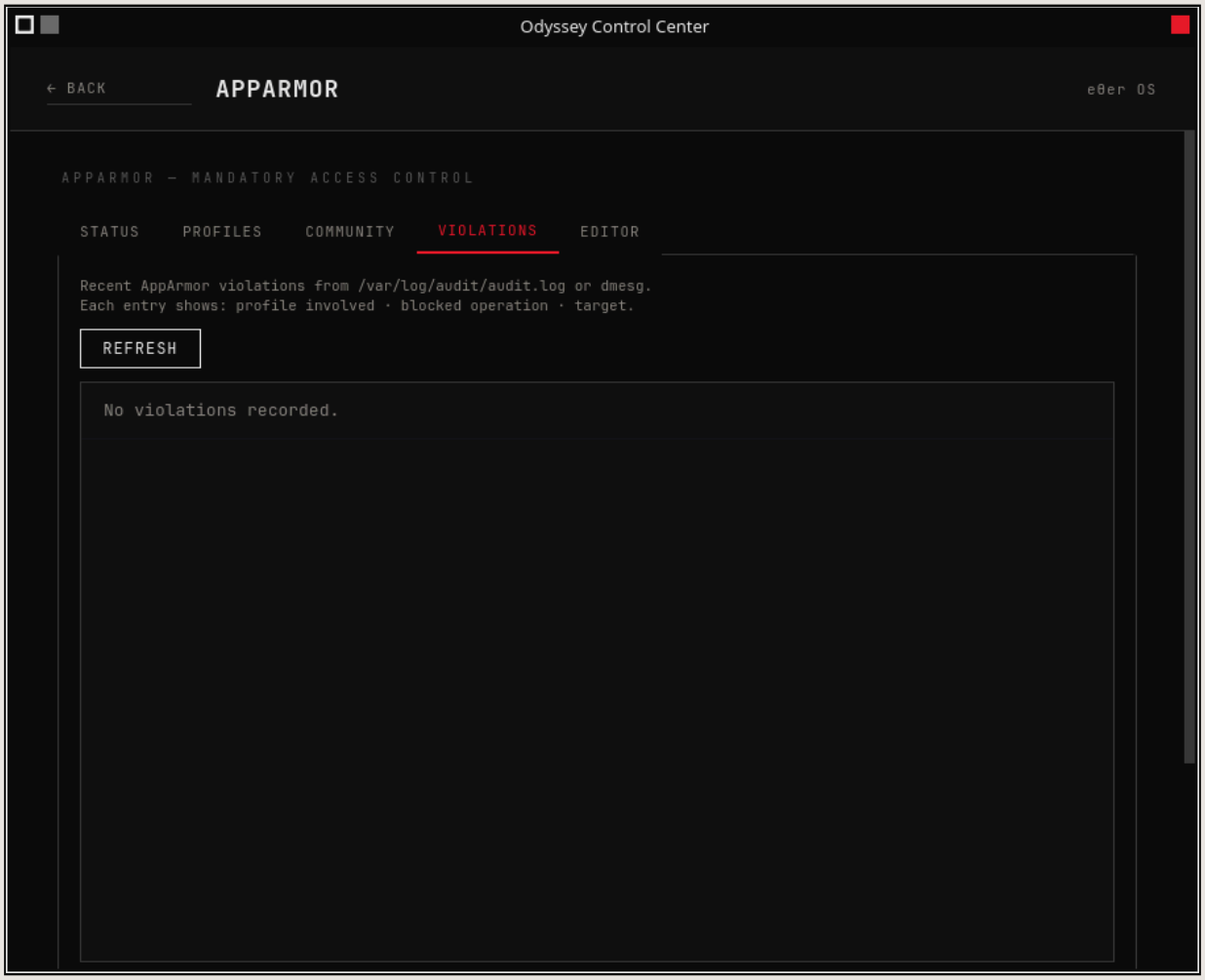

FIG · VIOLATIONS टैब: “No violations recorded”, एक REFRESH बटन के साथ, और एक नोट कि एंट्रीज़ /var/log/audit/audit.log या dmesg से आती हैं।

यह टैब सिस्टम का AppArmor ऑडिट लॉग पढ़ता है और दिखाता है कि किसी प्रोफ़ाइल ने **क्या ब्लॉक किया होता** (complain मोड में) या **क्या ब्लॉक किया** (enforce मोड में) — हर एंट्री प्रोफ़ाइल, ब्लॉक की गई कार्रवाई, और लक्ष्य फ़ाइल या संसाधन बताती है।

एक स्वस्थ सिस्टम पर ख़ाली सूची अपेक्षित नज़ारा है

स्क्रीनशॉट बिल्कुल वही दिखाता है जो आप आमतौर पर देखेंगे: कोई उल्लंघन नहीं। यह कोई बग नहीं है और न ही यह संकेत है कि पैनल सही से काम नहीं कर रहा — इसका मतलब है कि आख़िरी बार लॉग जाँचे जाने से हर प्रोफ़ाइल वाला प्रोग्राम अनुमत सीमाओं के भीतर ही रहा है। जब कोई प्रोफ़ाइल वाली चीज़ ऐसा कुछ करने की कोशिश करती है जिसकी उसकी प्रोफ़ाइल इजाज़त नहीं देती, तो उल्लंघन यहाँ दिखने लगते हैं; संबंधित प्रोग्राम को इस्तेमाल करने जाएँ, वापस आएँ, और REFRESH दबाएँ।

जब एंट्रीज़ दिखने लगें, तो उनमें से किसी एक पर डबल-क्लिक करना EDITOR में जाने का तेज़ रास्ता है, जहाँ वह ठीक वही प्रोफ़ाइल पहले से खुली होती है — नीचे देखें।

6.5 दृश्य गाइड — EDITOR

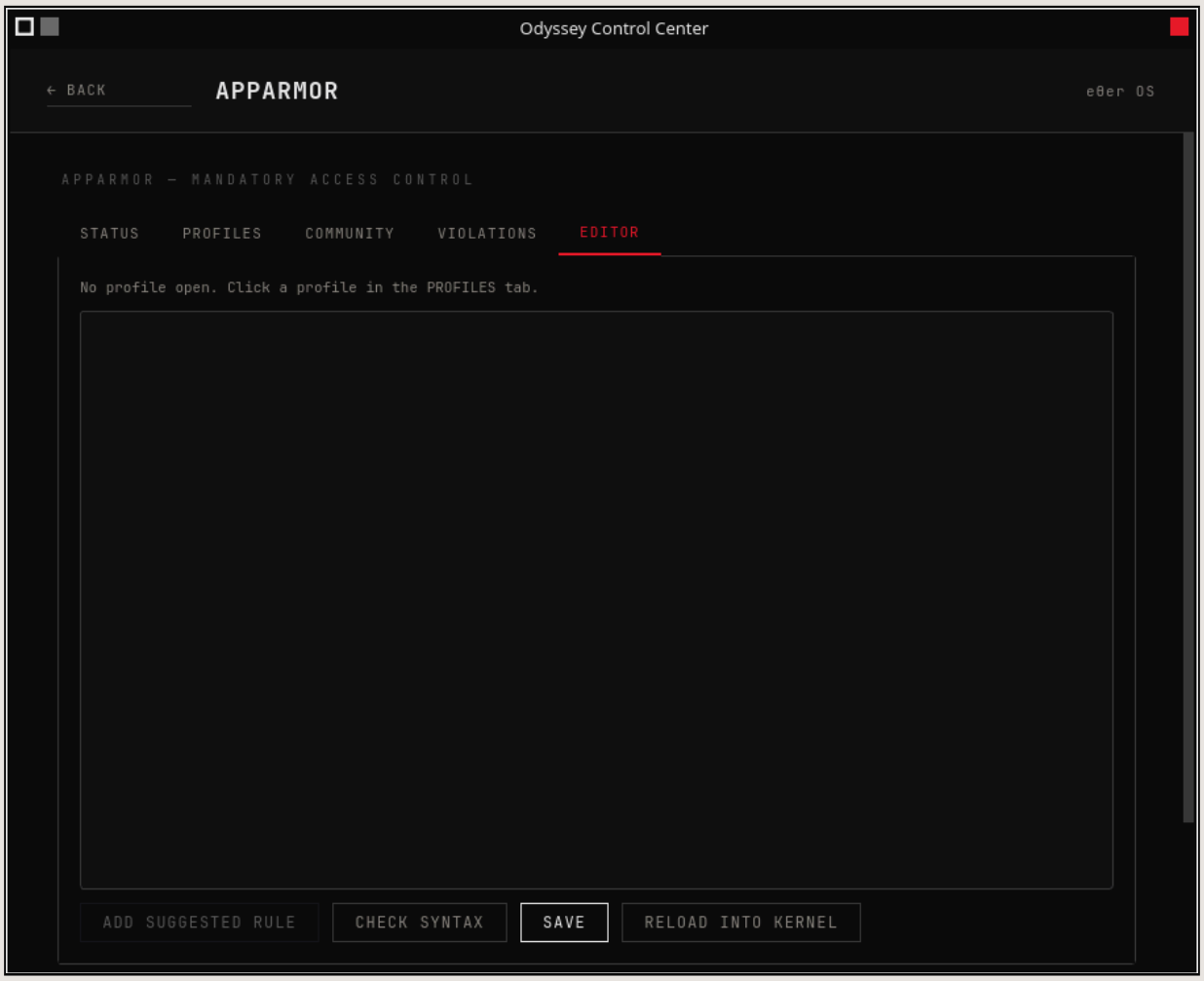

FIG · EDITOR टैब: “No profile open. Click a profile in the PROFILES tab”, एक ख़ाली टेक्स्ट एरिया, और चार बटन — ADD SUGGESTED RULE, CHECK SYNTAX, SAVE, RELOAD INTO KERNEL — SAVE को छोड़कर सब बंद।

एडिटर जानबूझकर **ख़ाली** खुलता है, जैसा दिखाया गया है — यह हमेशा एक बार में सिर्फ़ एक प्रोफ़ाइल संपादित करता है, और आप तय करते हैं कि कौन-सी PROFILES से (किसी लाइन पर क्लिक करके) या VIOLATIONS से (किसी एंट्री पर डबल-क्लिक करके)। यह ग़लती से ग़लत फ़ाइल संपादित कर देने की आसान चूक को रोकता है।

बटन	क्या करता है
ADD SUGGESTED RULE	सिर्फ़ तभी सक्रिय जब आप किसी VIOLATIONS एंट्री से आए हों — वह सटीक नियम-लाइन जोड़ता है जो दर्ज की गई कार्रवाई की इजाज़त दे देती

बटन	क्या करता है
CHECK SYNTAX	बिना प्रोफ़ाइल लोड किए उसे जाँचता है — मायने रखने से पहले टाइपिंग की ग़लती पकड़ लेता है
SAVE	आपके संपादन डिस्क पर मौजूद प्रोफ़ाइल फ़ाइल में लिख देता है
RELOAD INTO KERNEL	सेव किए गए संपादन को तुरंत असरदार बना देता है, अगले बूट या सेवा के फिर से शुरू होने का इंतज़ार किए बिना

WHAT	PROFILES → किसी प्रोफ़ाइल के नाम पर क्लिक करें → EDITOR उसे खोलता है → एक संपादन करें → CHECK SYNTAX → SAVE → RELOAD INTO KERNEL।
WHY	यही कन्फ़ाइनमेंट का पूरा चक्र एक लाइन में है: complain मोड में देखें (VIOLATIONS), जो असल में कमी है उसे ठीक करें (EDITOR), और तभी उस प्रोफ़ाइल को enforce में बदलें (PROFILES)।
RESULT	प्रोफ़ाइल बिल्कुल वैसे ही व्यवहार करती है जैसे संपादित की गई थी, लाइव, बिना रीबूट के।
RISK	किसी प्रोफ़ाइल को ग़लत तरीक़े से संपादित करना बाद में enforce के तहत किसी वैध चीज़ को ब्लॉक कर सकता है — CHECK SYNTAX SAVE से पहले ग़लत फ़ॉर्मेट वाले नियम पकड़ लेता है, लेकिन तार्किक ग़लतियाँ नहीं पकड़ता। अगर कोई प्रोफ़ाइल जिसे आपने संपादित किया है enforce के तहत परेशानी पैदा करने लगे, तो उसे PROFILES से complain में वापस लाएँ और VIOLATIONS दोबारा जाँचें।

▸ **ADVANCED**

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

6.6 Advanced

कमांड-लाइन के बराबर विकल्प

```
$ sudo aa-status
# टर्मिनल से, PROFILES टैब के हेडर जैसी ही गिनतियाँ
$ sudo aa-enforce /etc/apparmor.d/PROFILE_FILE
$ sudo aa-complain /etc/apparmor.d/PROFILE_FILE
$ sudo aa-disable /etc/apparmor.d/PROFILE_FILE
# हाथ से किए गए संपादन के बाद RELOAD INTO KERNEL के बराबर
$ sudo apparmor_parser -r /etc/apparmor.d/PROFILE_FILE
```

कच्चा ऑडिट लॉग पढ़ना

```
$ sudo grep apparmor /var/log/audit/audit.log | tail -20
# या, अगर कोई समर्पित ऑडिट डेमन नहीं चल रहा है:
$ sudo dmesg | grep -i apparmor
```

हाथ से किसी लाइन को पढ़ते समय जानने लायक उपयोगी फ़ील्ड्स: `apparmor="DENIED"` (enforce) या `apparmor="ALLOWED"` (complain, सिर्फ़ दर्ज), `operation=` (क्या करने की कोशिश की गई), `profile=` (किस प्रोफ़ाइल ने फ़ैसला लिया — यह वही है जिसे EDITOR खोलता है), `name=` (पथ या संसाधन), `comm=` (प्रोसेस)।

बल्क ALL→ENFORCE जोखिम भरा क्यों है

किसी प्रोफ़ाइल के VIOLATIONS लॉग के कुछ समय तक शांत रहने के बाद उसे अलग-अलग बढ़ाना सुरक्षित तरीक़ा है — आप प्रोफ़ाइल-दर-प्रोफ़ाइल जानते हैं कि सामान्य इस्तेमाल इसे ट्रिगर नहीं करता। ALL→ENFORCE सभी प्रोफ़ाइलों के लिए एक साथ इस निगरानी अवधि को छोड़ देता है, उन प्रोफ़ाइलों समेत जिन्हें आपने शायद अभी-अभी COMMUNITY से इंस्टॉल किया हो और कभी सच में इस्तेमाल

न किया हो। जिस प्रोफ़ाइल को जिए बिना ही लागू कर दिया जाए, वह ठीक उसी पल किसी वैध सुविधा को ब्लॉक कर सकती है जब आप उसे पहली बार इस्तेमाल करते हैं। एक-एक करके आगे बढ़ने के चक्र को प्राथमिकता दें; बल्क क्रियाएँ ALL→COMPLAIN के लिए बचाकर रखें (हमेशा सुरक्षित — सिर्फ़ ढील देता है) या समस्या सुलझाते समय ALL→DISABLE के लिए।

इस पैनल द्वारा छुई जाने वाली फ़ाइलें

पथ	भूमिका
`/etc/apparmor.d/`	हर सीमित प्रोग्राम के लिए एक प्रोफ़ाइल फ़ाइल — वही जो PROFILES सूचीबद्ध करता है और EDITOR खोलता है
`/etc/default/apparmor`	बूट पर डिफ़ॉल्ट मोड (`APPARMOR=complain`) और enable/disable स्थिति
`/etc/default/grub`	वे कर्नेल पैरामीटर `apparmor=1 security=apparmor` जिन्हें STATUS जाँचता है

CHAPTER 7

Privacy

दो स्वतंत्र टूल, हर एक का अपना install → enable → running वाला तीन-चरणों का सिलसिला, और दोनों के चालू हो जाने पर उनके बीच एक पुल।

7.1 दृश्य गाइड — SearXNG, चरण-दर-चरण

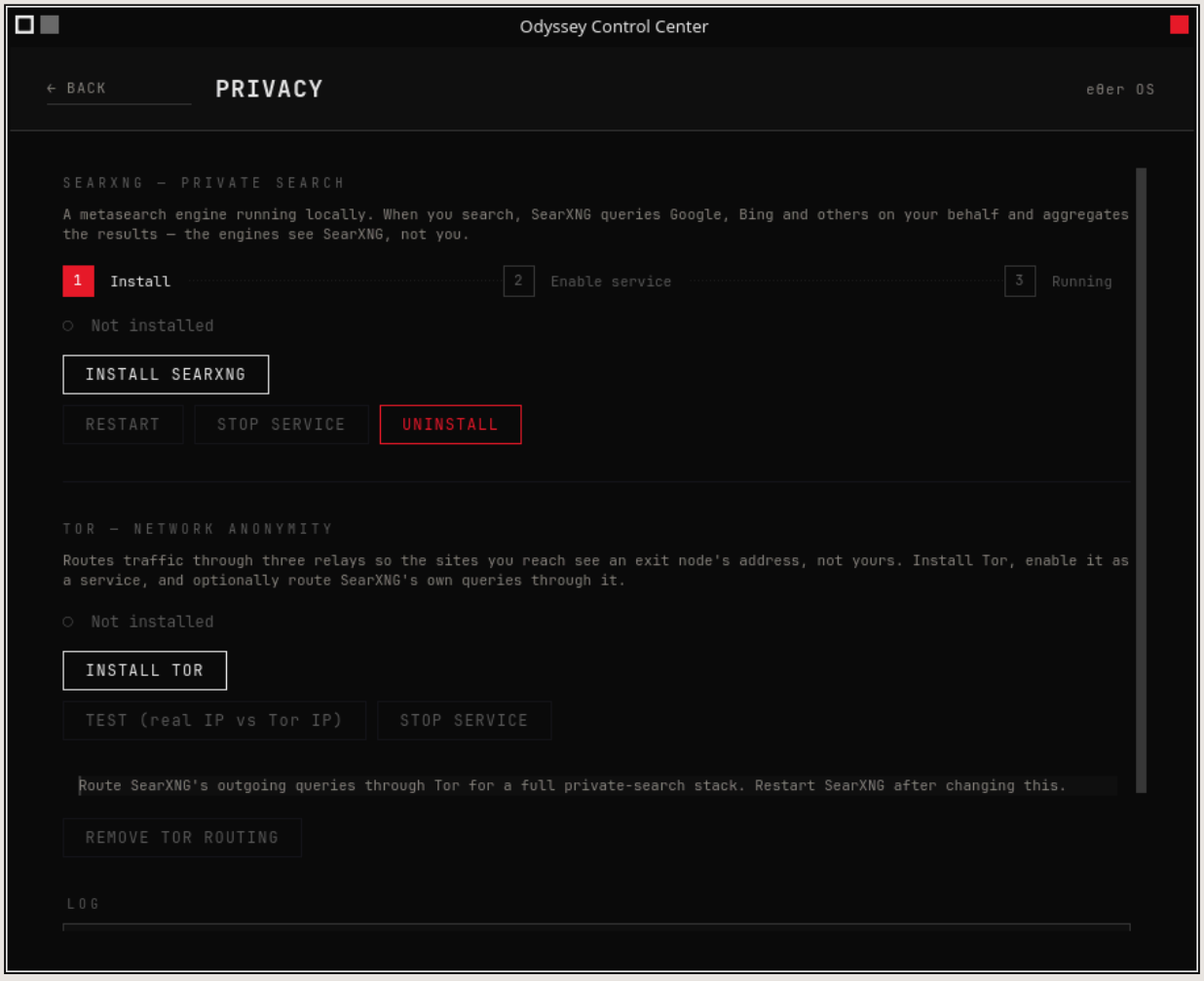

FIG · शांत हालत में पैनल: SearXNG का ट्रैकर चरण 1 “Install” पर, स्टेटस “Not installed”, सिर्फ़ एक बटन INSTALL SEARXNG; Tor का ट्रैकर भी “Not installed” दिखाता है नीचे INSTALL TOR के साथ।

SearXNG एक मेटा-सर्च इंजन है जो आपकी अपनी मशीन पर स्थानीय रूप से चलता है — जब आप खोजते हैं, तो आपकी ओर से Google, Bing और दूसरों से पूछने वाला SearXNG होता है, इसलिए वे इंजन SearXNG को अनुरोध करते हुए देखते हैं, कभी सीधे आपको नहीं।

1 **INSTALL SEARXNG**

क्लिक करें। लॉग असली इंस्टॉलेशन को चलते हुए दिखाता है — सोर्स कोड क्लोन करना, एक सिस्टम यूज़र बनाना, एक Python वर्चुअल एनवायरनमेंट सेट करना।

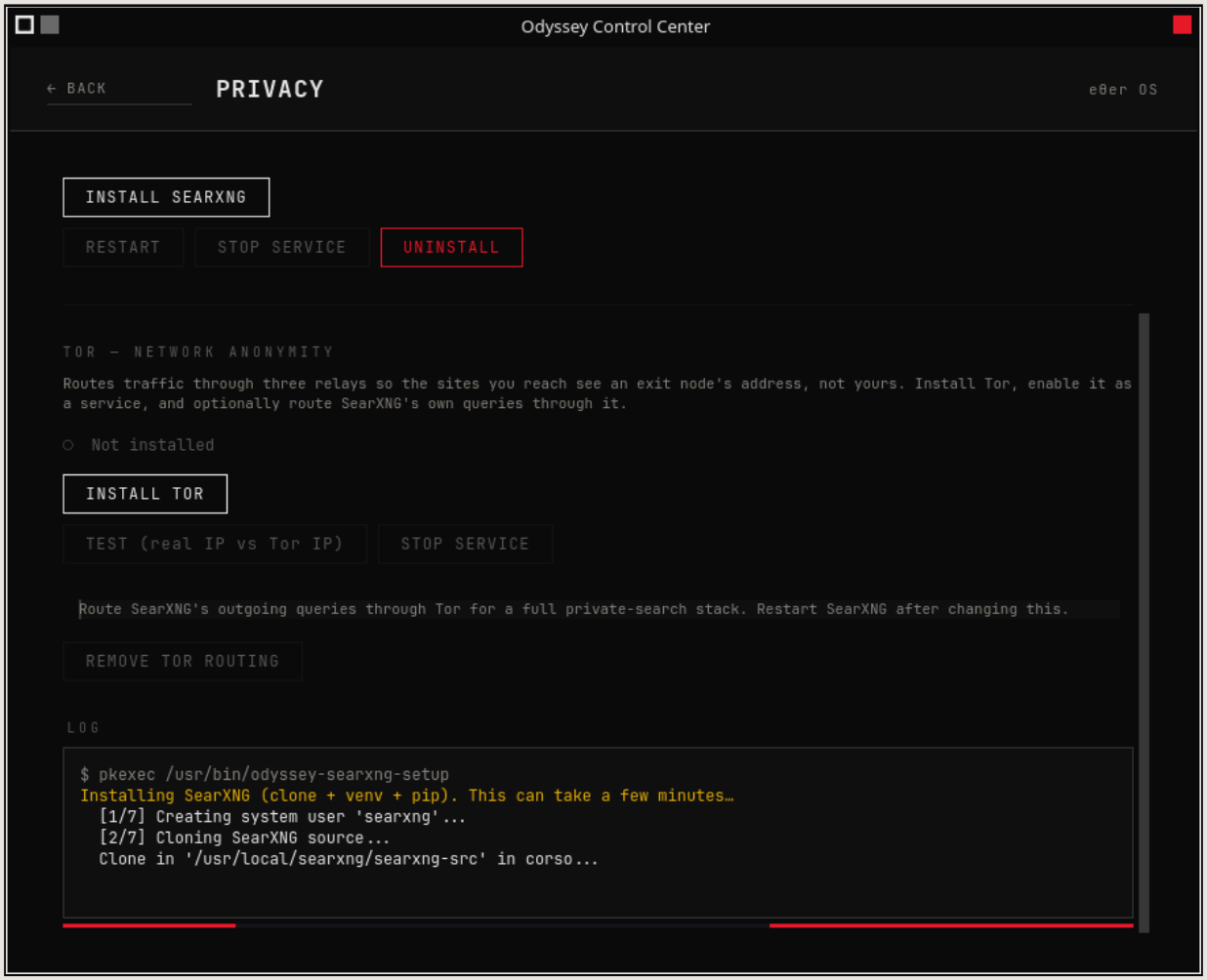

FIG · इंस्टॉलेशन के बीच में: लॉग असली आउटपुट भेज रहा है — “Installing SearXNG (clone + venv + pip). This can take a few minutes…”, चरण 1/7 और 2/7 दिखाई दे रहे हैं।

2 ENABLE SERVICE

इंस्टॉल होने के बाद, ट्रैकर चरण 2 पर जाता है और स्टेटस लाइन पीली हो जाती है: “Installed — service not enabled”। अब सिर्फ़ एक बटन: ENABLE SERVICE।

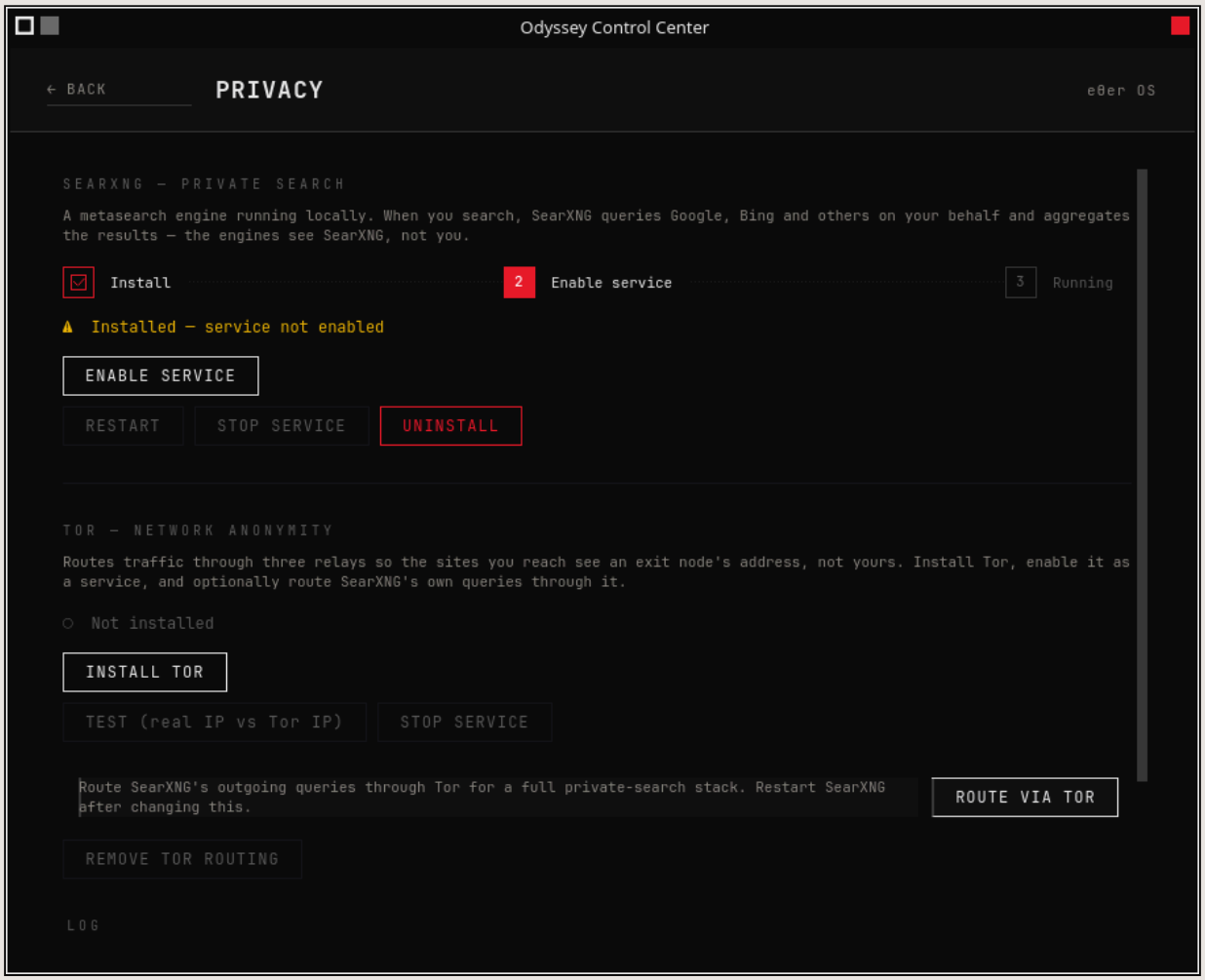

FIG · चरण 2 सक्रिय: “Installed — service not enabled”, ENABLE SERVICE अकेला सक्रिय बटन, नीचे RESTART/STOP SERVICE/UNINSTALL अभी भी बंद।

3 Running

चालू करने के बाद, ट्रैकर चरण 3 पर पहुँचता है और नियंत्रण की पूरी लाइन सक्रिय हो जाती है। एक बार चलने लगे, तो **OPEN IN BROWSER** आपको सीधे वहाँ ले जाता है — SearXNG डिफ़ॉल्ट रूप से `http://localhost:8888` पर सुनता है।

WHAT	Privacy → INSTALL SEARXNG → ENABLE SERVICE → OPEN IN BROWSER।
WHY	हर व्यावसायिक सर्च इंजन समय के साथ आपकी खोजों से एक प्रोफ़ाइल बनाता है; एक स्थानीय मेटा-सर्च इंस्टेंस का मतलब है कि सिर्फ़ आप ही कभी वह बिना-जोड़ी गई खोज देखते हैं।
RESULT	आपकी अपनी मशीन पर चलने वाला एक निजी सर्च इंजन, आपके ब्राउज़र में पहुँच योग्य, आपकी ओर से बड़े सर्च इंजनों से पूछता हुआ बिना आपको सीधे उनके सामने रखे।
RISK	कम — यह किसी और की तरह एक स्थानीय वेब सेवा है। अगर यह ग़लत व्यवहार करे तो RESTART, बिना हटाए रोकने के लिए STOP SERVICE, पूरी तरह हटाने के लिए UNINSTALL।

7.2 दृश्य गाइड — Tor, चरण-दर-चरण

वही तीन-चरण का ढाँचा, स्वतंत्र ट्रैकर, ठीक SearXNG सेक्शन के नीचे।

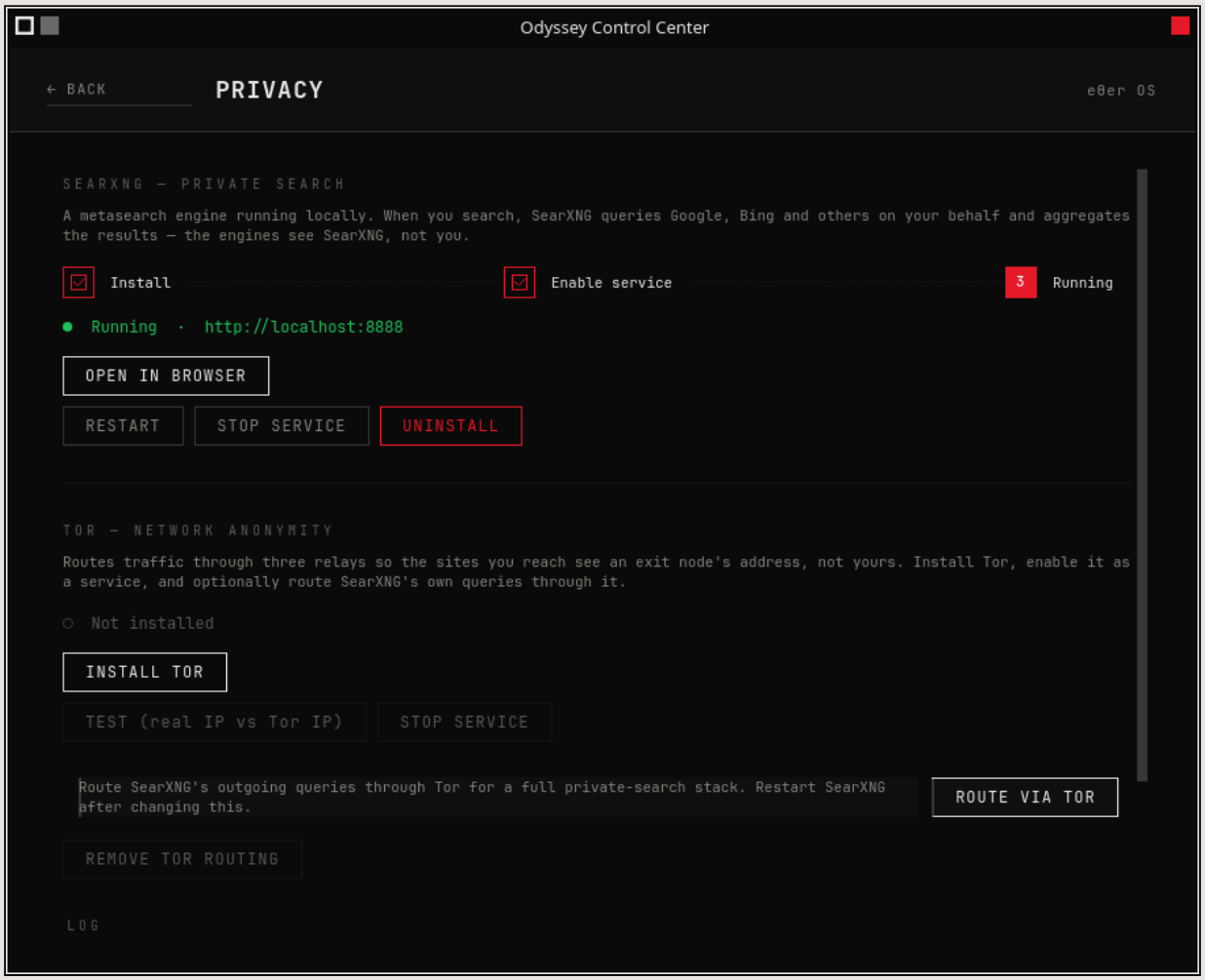

FIG · ऊपर SearXNG पूरी तरह चल रहा है; नीचे Tor सेक्शन अभी भी “Not installed” पर है, INSTALL TOR अकेला सक्रिय बटन।

Tor आपके ट्रैफ़िक को उसके गंतव्य तक पहुँचने से पहले तीन रिले से गुज़ारता है, इसलिए जिस साइट पर आप जाते हैं वह बाहर निकलने वाले रिले का पता देखती है, कभी आपका नहीं।

1 **INSTALL TOR**

लॉग पैकेज इंस्टॉलेशन की पुष्टि करता है और सेवा को runit से जुड़ा हुआ दिखाता है।

2 **ENABLE SERVICE**

स्टेटस “Installed — service not enabled” में बदल जाती है — SearXNG जैसा ही पैटर्न।

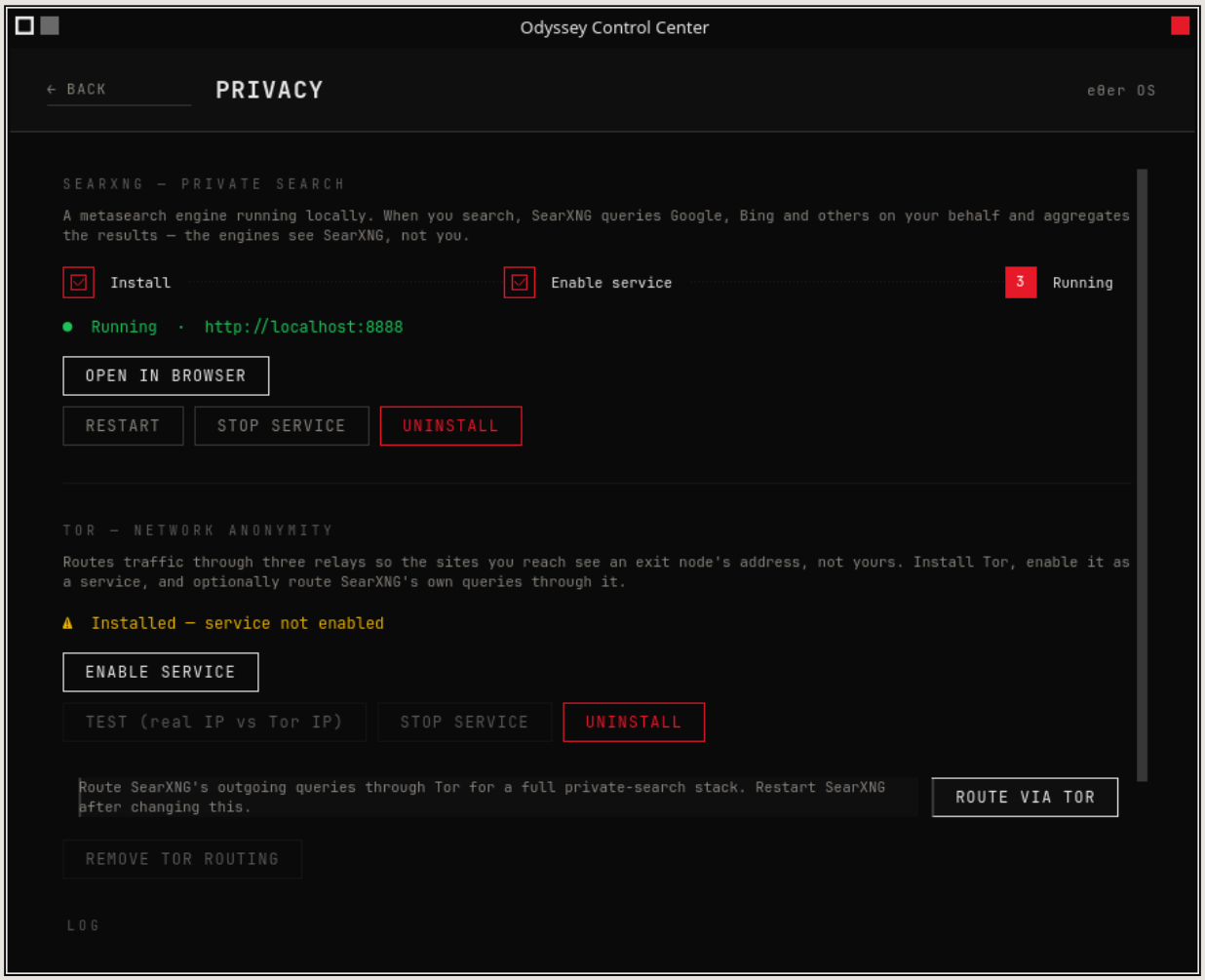

FIG · चरण 2 पर Tor: “Installed — service not enabled”, ENABLE SERVICE सक्रिय, TEST और STOP SERVICE अभी भी बंद।

3 Running, और इसकी जाँच करें

चालू करने के बाद, एक समर्पित **TEST (real IP vs Tor IP)** बटन दिखाई देता है — इसका इस्तेमाल करें। यह आपके असली पते की तुलना उस पते से करता है जो Tor दिखा रहा है, पास-पास, ताकि आपके पास यह असली सबूत हो कि यह काम कर गया, न कि सिर्फ़ एक स्टेटस लाइट जिस पर आपको बस भरोसा करना पड़ता है।

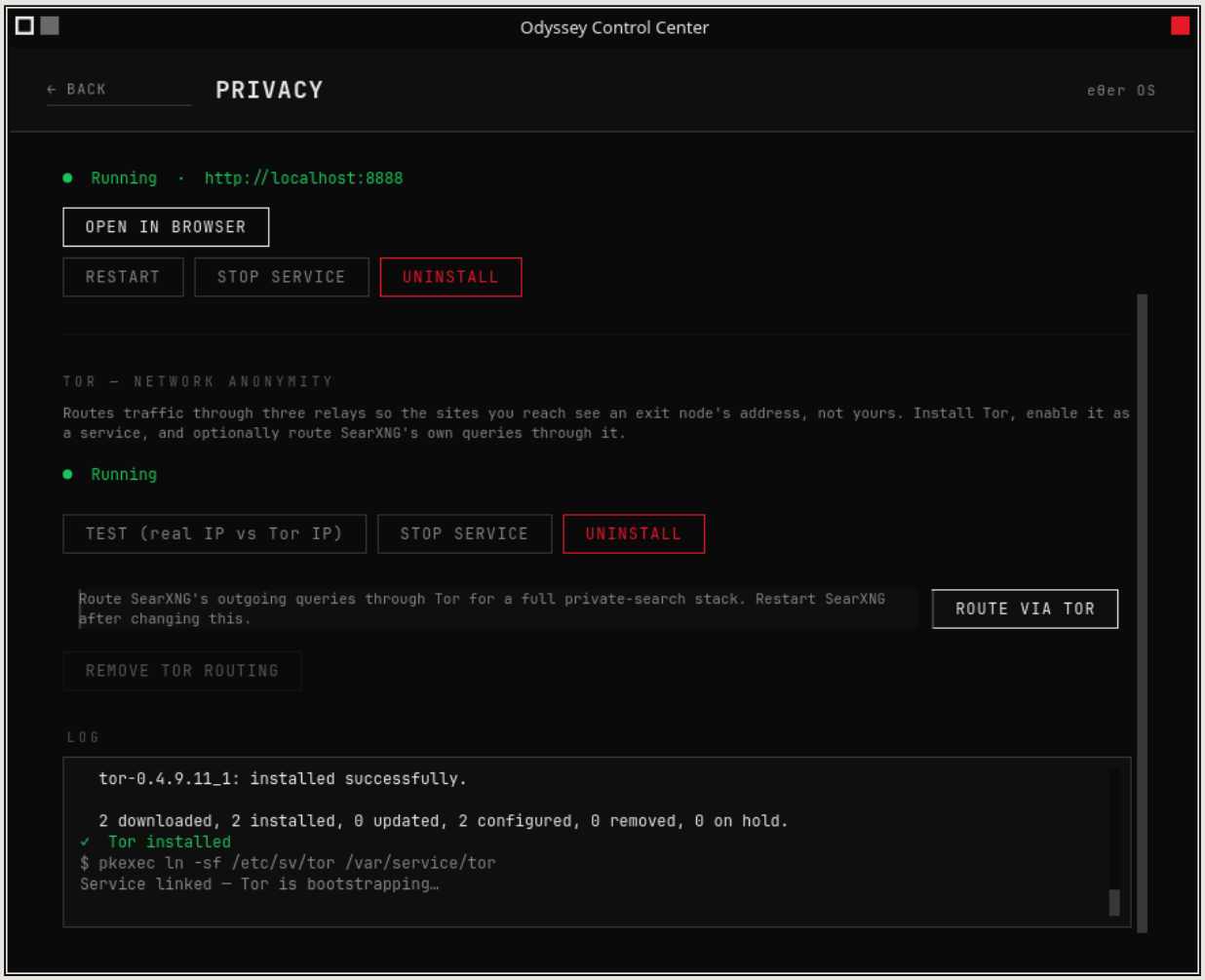

FIG · दोनों सेक्शन हरे और चल रहे: SearXNG localhost:8888 पर OPEN IN BROWSER के साथ, Tor चल रहा है TEST / STOP SERVICE / UNINSTALL सब सक्रिय के साथ।

WHAT	Privacy → INSTALL TOR → ENABLE SERVICE → पुष्टि के लिए TEST (real IP vs Tor IP)।
WHY	संवेदनशील ब्राउज़िंग को Tor के ज़रिए रूट करना आपके पते को उन साइटों से छिपाता है जिन पर आप जाते हैं — TEST आपको सबूत देता है, सिर्फ़ दावा नहीं।
RESULT	एक चल रही Tor सेवा जिसके ज़रिए दूसरे ऐप्लिकेशन रूट हो सकते हैं; अगर यह काम कर रहा है तो TEST दो सच में अलग पते दिखाता है।
RISK	Tor के ज़रिए ट्रैफ़िक साफ़ तौर पर धीमा होता है — यह गुमनामी की क़ीमत है। STOP SERVICE इसे रोकता है; UNINSTALL इसे हटा देता है।

7.3 दृश्य गाइड — दोनों को मिलाना

एक बार दोनों चलने लगें, तो एक इनलाइन सूचना दिखती है जो SearXNG की **अपनी** क्वेरीज़ को भी Tor के ज़रिए रूट करने का सुझाव देती है — सिर्फ़ SearXNG को आपके अनुरोध नहीं, बल्कि आपकी ओर से Google और Bing को SearXNG के अनुरोध भी।

WHAT	Privacy → “Route SearXNG’s outgoing queries through Tor” सूचना → ROUTE VIA TOR → इसे लागू करने के लिए SearXNG सेक्शन में RESTART।
WHY	इसके बिना, SearXNG ख़ुद सीधे बड़े सर्च इंजनों से पूछता रहता है — आपकी तरफ़ से निजी, पर उनकी तरफ़ से दिखाई देता हुआ। SearXNG के अपने ट्रैफ़िक को Tor के ज़रिए रूट करना पूरी तरह निजी सर्च स्टैक के लिए इस कमी को पूरा करता है।
RESULT	SearXNG की बाहर जाने वाली क्वेरीज़, सिर्फ़ इसे भेजी गई आपकी क्वेरीज़ नहीं, Tor से होकर गुज़रती हैं।
RISK	Tor से रूट किए गए किसी भी ट्रैफ़िक जैसी ही वजह से नतीजे साफ़ तौर पर धीमे लोड होते हैं। REMOVE TOR ROUTING इसे उलट देता है, फिर से रीस्टार्ट के बाद।

▸ ADVANCED

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

7.4 Advanced

INSTALL SEARXNG असल में क्या चलाता है

```
$ pkexec /usr/bin/odyssey-searxng-setup
# एक समर्पित सिस्टम यूज़र 'searxng' बनाता है,
# सोर्स कोड को /usr/local/searxng/searxng-src में क्लोन करता है,
# एक Python venv बनाता है और pip से निर्भरताएँ इंस्टॉल करता है
```

सेवाएँ और पोर्ट, हाथ से

```
# SearXNG की अपनी runit सेवा
$ sudo sv status searxng
$ sudo sv restart searxng
# Tor की runit सेवा — पैनल का चालू करने वाला चरण शब्दशः यही है
$ sudo ln -sf /etc/sv/tor /var/service/tor
$ sudo sv status tor
# पुष्टि करें कि Tor असल में रूट कर रहा है
$ curl -socks5 localhost:9050 https://check.torproject.org/api/ip
```

SEARXNG को हाथ से TOR के ज़रिए रूट करना

Tor का SOCKS प्रॉक्सी डिफ़ॉल्ट रूप से `127.0.0.1:9050` पर सुनता है। SearXNG को इसके ज़रिए रूट करने का मतलब है उसके बाहर जाने वाले HTTP क्लाइंट को इस प्रॉक्सी की ओर इशारा करना — यह सेटिंग किसी सिस्टम-वाइड प्रॉक्सी वेरिएबल में नहीं, बल्कि SearXNG की अपनी `settings.yml` फ़ाइल में, बाहर जाने वाले अनुरोधों की कॉन्फ़िगरेशन के तहत रहती है। पैनल का ROUTE VIA TOR बटन इस फ़ाइल को सीधे संपादित करता है और सेवा को फिर से शुरू करता है; इसे हाथ से करने का मतलब है `/usr/local/searxng/searxng-src/searx/settings.yml` में वही `outgoing:` ब्लॉक ढूँढना और संपादित करना।

> **रूटिंग बदलाव के बाद SEARXNG को रीस्टार्ट की ज़रूरत क्यों पड़ती है**
>
> SearXNG अपनी प्रॉक्सी कॉन्फ़िगरेशन सिर्फ़ एक बार, शुरू होते समय पढ़ता है — यह ऐसी सेटिंग नहीं है जिसे चलते-चलते फिर से लोड किया जा सके। यह तब भी सच है जब आप इसे पैनल से बदलें या हाथ से, यही वजह है कि दोनों रास्ते एक ही RESTART चरण पर ख़त्म होते हैं।

इस पैनल द्वारा छुई जाने वाली फ़ाइलें

पथ	भूमिका
`/usr/local/searxng/searxng-src/`	SearXNG की अपनी इंस्टॉलेशन — सोर्स कोड, venv, कॉन्फ़िगरेशन
`/etc/sv/searxng`, `/etc/sv/tor`	runit सेवा परिभाषाएँ
`/var/service/`	जहाँ किसी सेवा को चालू करना उसे चल रहे सुपरविज़न ट्री से जोड़ देता है

CHAPTER 8

Containers

शुरू से अंत तक कंटेनर प्रबंधन: इंजन सेटअप, चल रहे कंटेनरों की एक सूची, शुरुआत के लिए एक चुनिंदा कैटलॉग, जो कैटलॉग में नहीं उसके लिए कच्चे कमांड, और डिस्क इस्तेमाल — सब एक ही स्क्रॉल होने वाले पैनल में।

8.1 दृश्य गाइड — इंजन

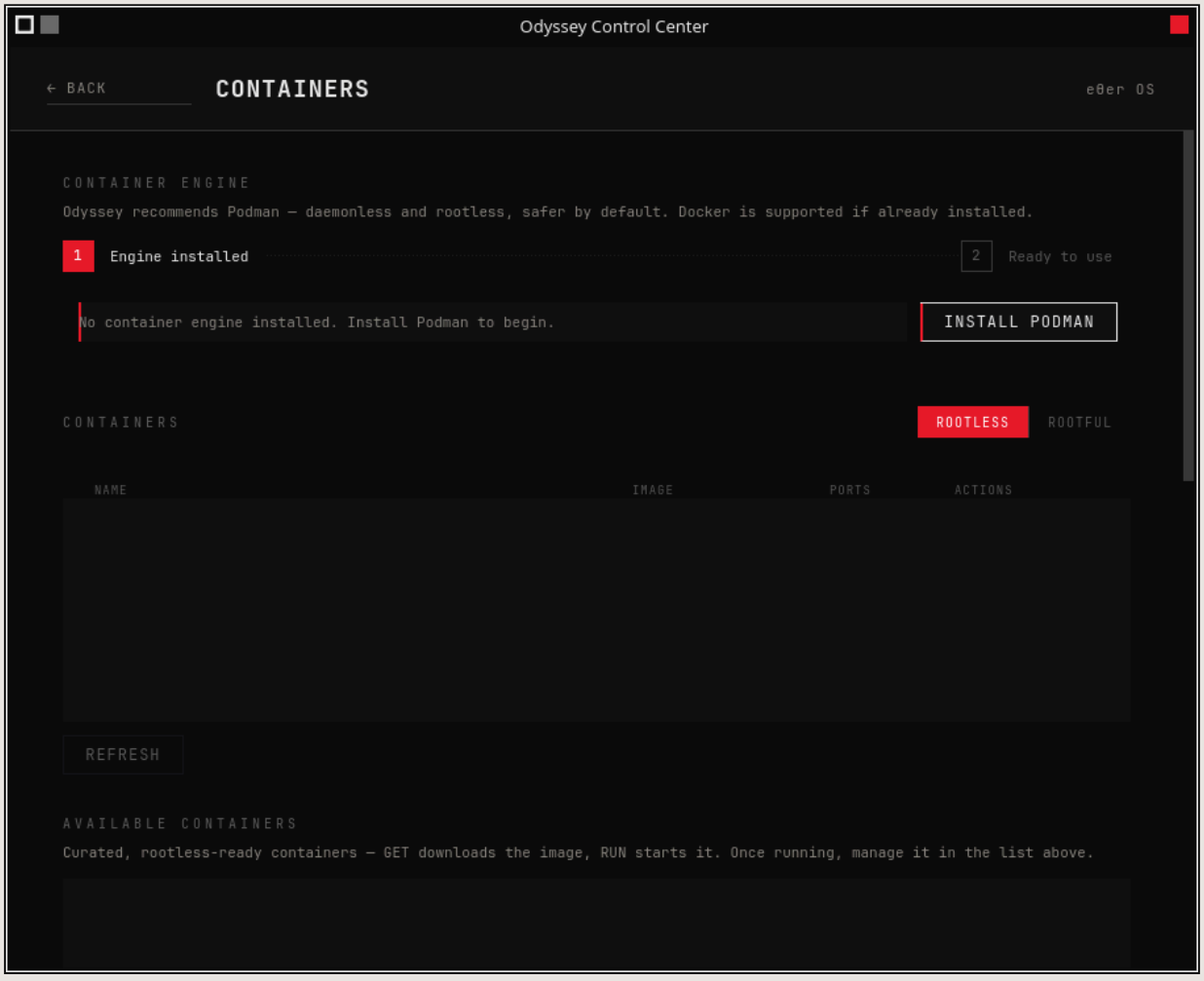

FIG · 2 में से 1 चरण: “No container engine installed. Install Podman to begin”, INSTALL PODMAN अकेली सक्रिय कार्रवाई के रूप में। नीचे Containers list और Available containers सेक्शन दिखाई देते हैं लेकिन ख़ाली हालत में।

Odyssey **Podman** की सलाह देता है — बिना डेमन के और rootless, जिसका मतलब है कि कंटेनर बिना किसी स्थायी बैकग्राउंड सेवा के चलते हैं और, डिफ़ॉल्ट रूप से, बिना root विशेषाधिकार के। अगर आप पहले से **Docker** इस्तेमाल करते हैं तो उसे भी समर्थन मिलता है।

1 **INSTALL PODMAN**

लॉग असली पैकेज इंस्टॉलेशन दिखाता है।

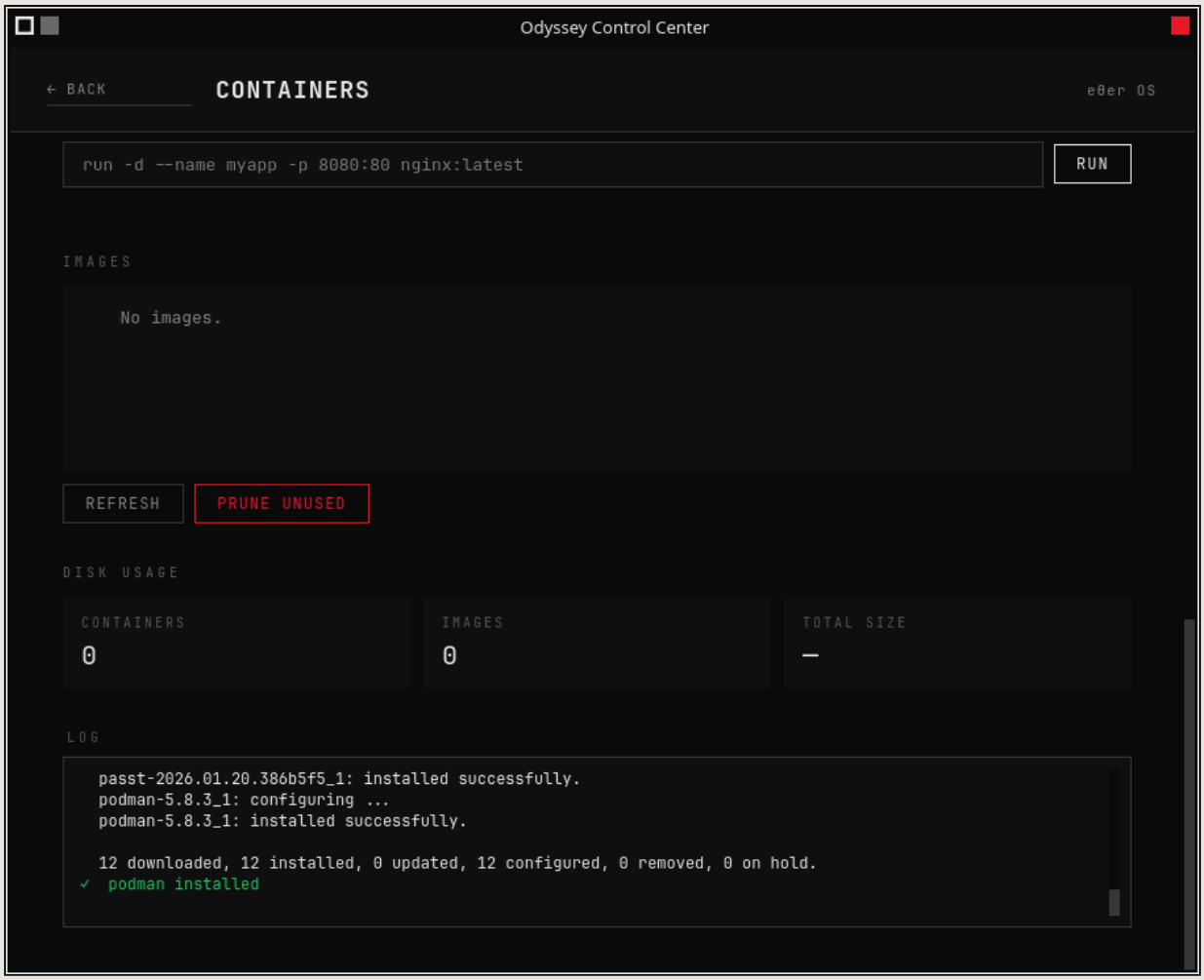

FIG · इंस्टॉलेशन के बाद: “Engine installed” लाल रंग में टिक किया गया, चरण 2 “Ready to use”, जानकारी की लाइन कहती है “podman version 5.8.3 · Podman installed · Docker not installed”, और एक साथ Docker भी चाहने वालों के लिए INSTALL DOCKER TOO बटन।

2 Ready to use

इंजन सक्रिय है। अगर आप दोनों इंजन साथ- साथ चाहते हैं तो **INSTALL DOCKER TOO** उपलब्ध रहता है — वे बिना किसी टकराव के साथ मौजूद रह सकते हैं।

8.2 दृश्य गाइड — rootless बनाम rootful, और ख़ाली हालत

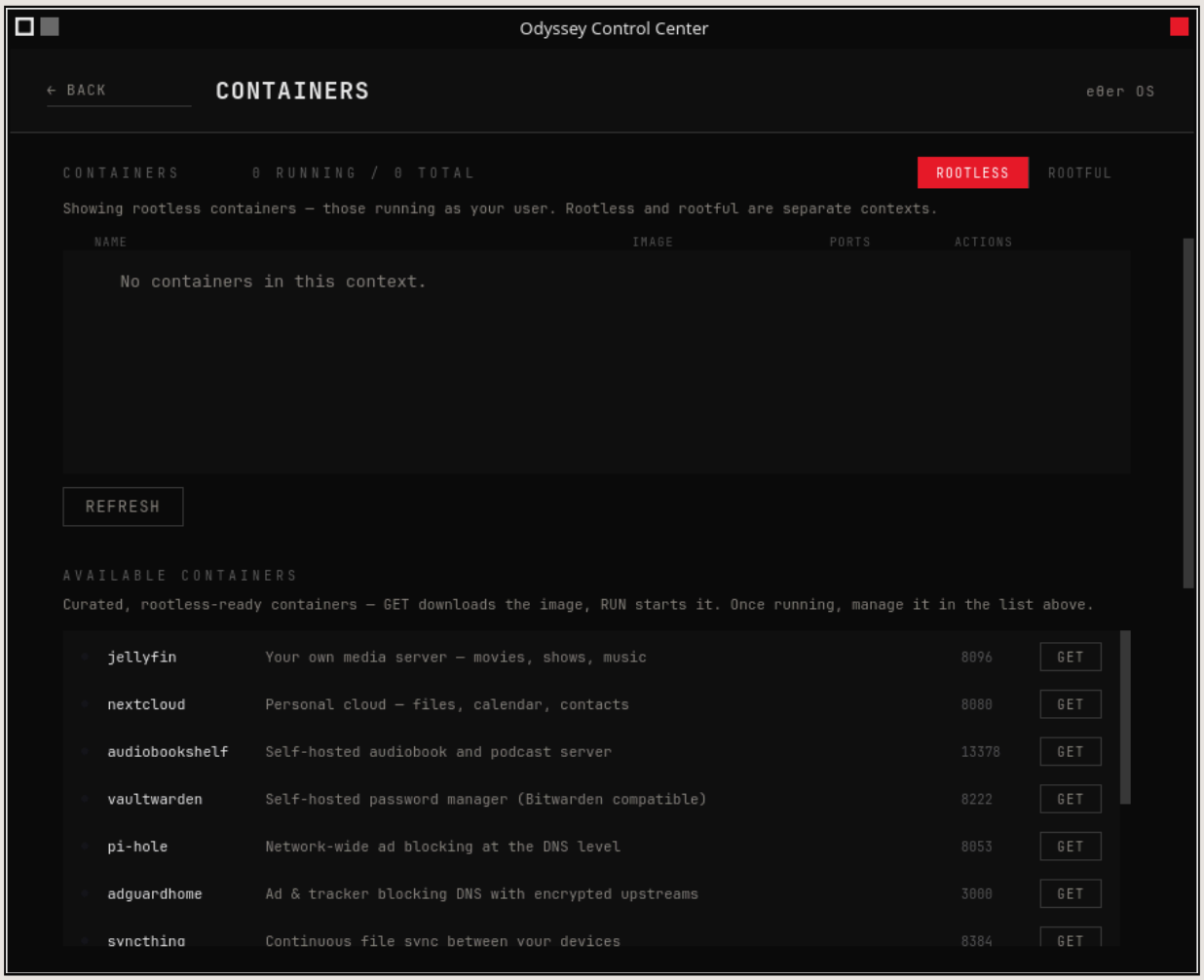

FIG · ROOTFUL के बगल में ROOTLESS चुना गया (लाल रंग में उभरा हुआ), "0 RUNNING / 0 TOTAL", और NAME / IMAGE / PORTS / ACTIONS कॉलम हेडिंग के नीचे "No containers in this context."।

ऊपर का **ROOTLESS / ROOTFUL** स्विच बदलता है कि आप कौन-सा संदर्भ देख और प्रबंधित कर रहे हैं — दोनों पूरी तरह अलग-अलग कंटेनर नेमस्पेस हैं, किसी मिली-जुली सूची पर फ़िल्टर नहीं। Odyssey डिफ़ॉल्ट रूप से ROOTLESS दिखाता है, और यह डिफ़ॉल्ट जानबूझकर है:

> **दोनों में से कौन-सा इस्तेमाल करना चाहिए**
>
> Rootless ज़्यादा सुरक्षित विकल्प है और पैनल आपको उसी की ओर मार्गदर्शित करता है: rootless मोड में कन्फ़ाइनमेंट से भाग निकला कंटेनर वैसे भी ऐसा कुछ नहीं कर सकता जो आपका अपना यूज़र अकाउंट पहले से न कर सकता हो। Rootful उन ख़ास मामलों के लिए है जिन्हें इसकी ज़रूरत है — कुछ कम-नंबर वाले नेटवर्क पोर्ट, कुछ हार्डवेयर एक्सेस — और यह अपवाद होना चाहिए, आदत नहीं।

8.3 दृश्य गाइड — चुनिंदा कैटलॉग, शुरू से अंत तक

Available containers तक स्क्रॉल करें — पहले से तैयार एक सूची ताकि पैनल कभी एक ख़ाली डिब्बा न बने जो इंतज़ार कर रहा हो कि आप इमेज के नाम पहले से याद रखें। हर लाइन कंटेनर, एक लाइन का विवरण, और वह पोर्ट दिखाती है जिसका वह इस्तेमाल करता है।

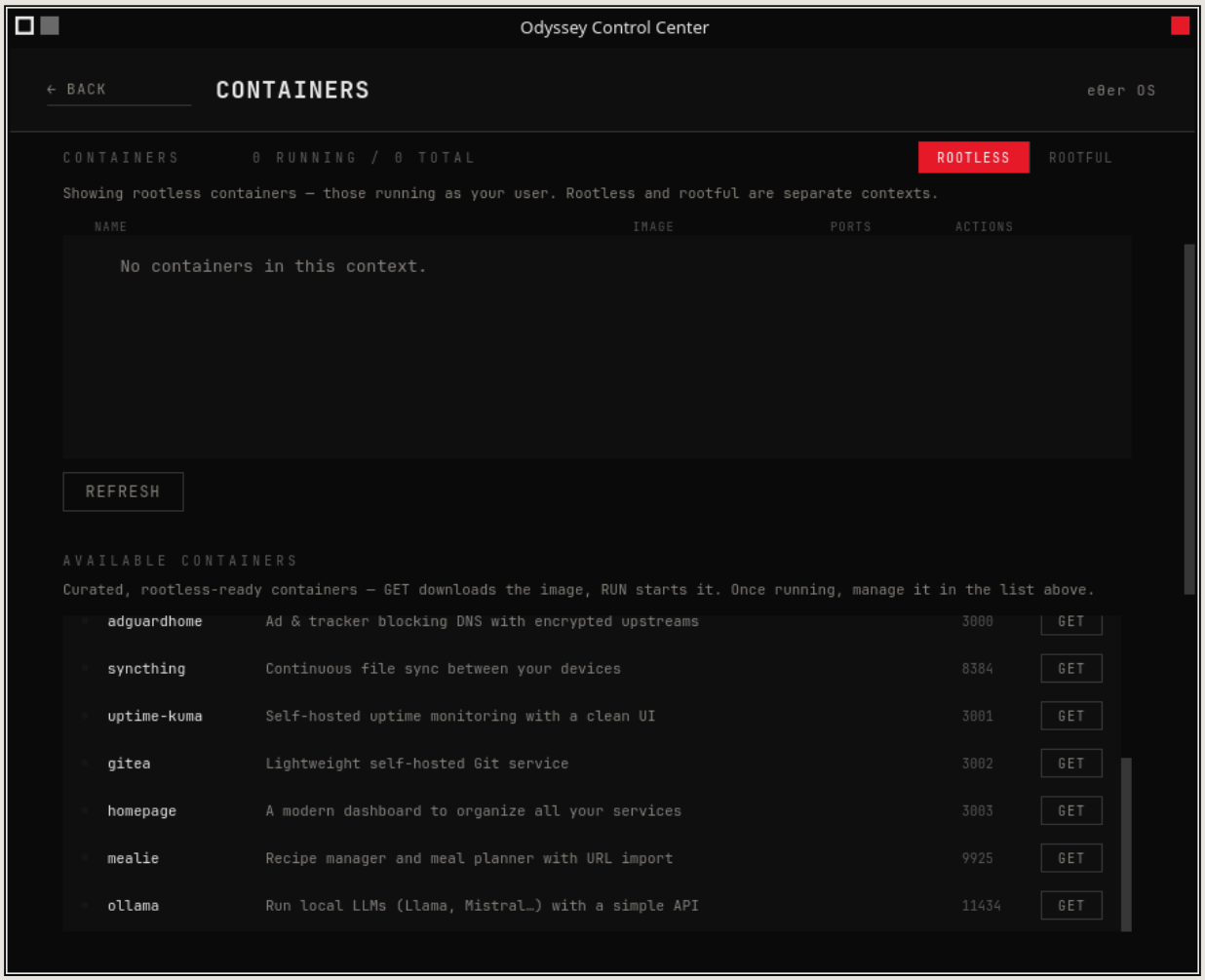

FIG · कैटलॉग की सूची: adguardhome (पोर्ट 3000), syncthing (8384), uptime-kuma (3001), gitea (3002), homepage (3003) — हर लाइन में एक GET बटन।

यह पोर्ट नंबर वही है जिस पर आप सेवा चलने के बाद उस तक पहुँचेंगे — उदाहरण के लिए `homepage` और `3003`, एक बार सक्रिय होने पर, मतलब है `http://localhost:3003`।

1 GET

`homepage` पर GET पर क्लिक करें। यह सिर्फ़ कंटेनर की इमेज डाउनलोड करता है — अभी कुछ भी नहीं चल रहा।

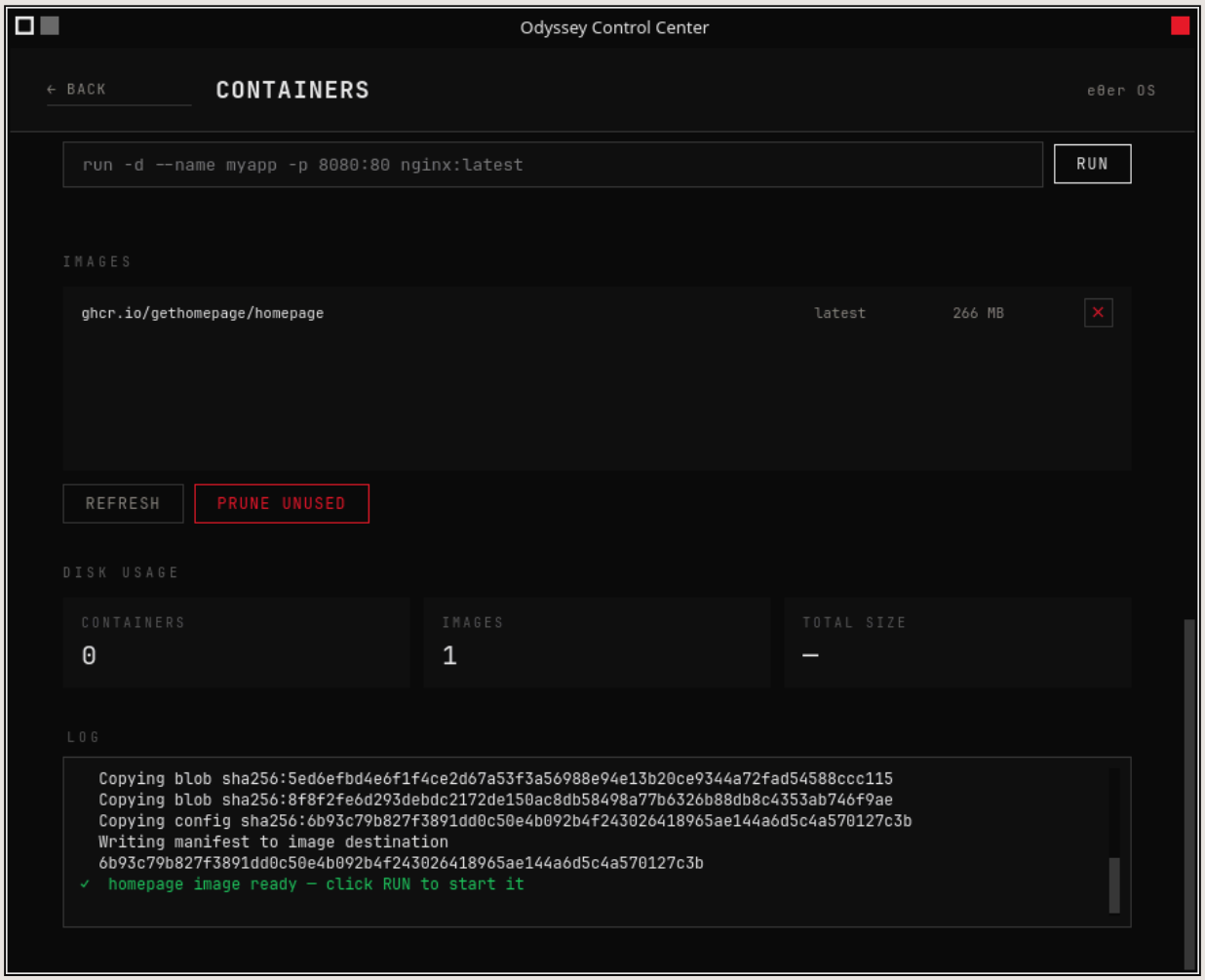

FIG · लॉग असली इमेज डाउनलोड भेज रहा है: “Copying blob sha256:5ed6…”, “Writing manifest to image destination”, हरे रंग में ख़त्म — “homepage image ready — click RUN to start it”।

2 RUN

जहाँ GET बटन था वह लाइन अब उसकी जगह RUN दिखाती है — कंटेनर को असल में शुरू करने के लिए उस पर क्लिक करें।

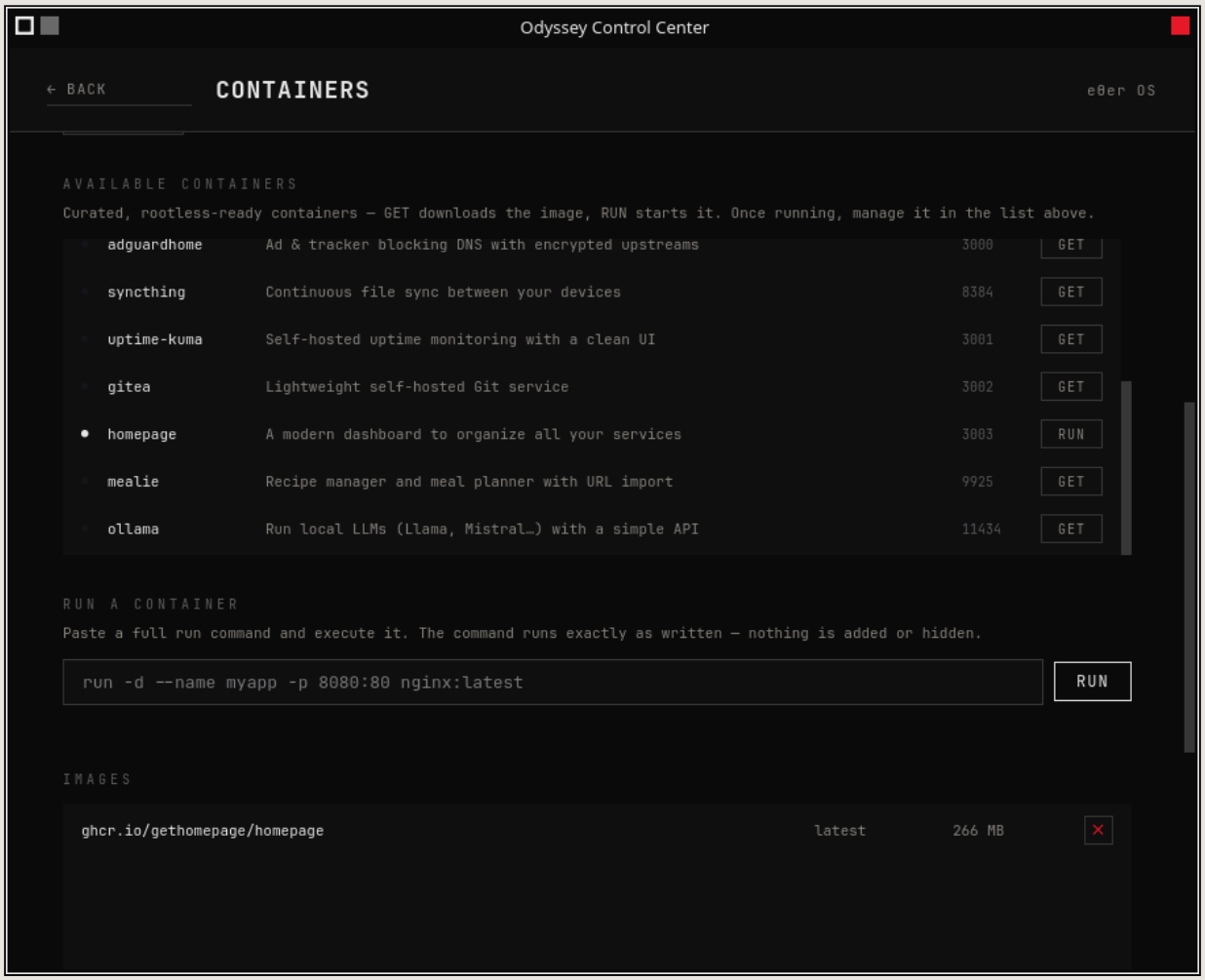

FIG · homepage अब ऊपर CONTAINERS सेक्शन में सूचीबद्ध है, हरे रंग की बिंदी, इमेज पथ ghcr.io/gethomepage/homepage, पोर्ट "0.0.0.0:3003→…", अपनी ख़ुद की एक्शन आइकन की लाइन के साथ; नीचे Available Containers में अभी भी दिखाई देता है, एक और इंस्टेंस शुरू करने के लिए नए RUN बटन के साथ।

एक बार चलने लगे, तो homepage दो जगह एक साथ दिखता है: ऊपर **Containers** सूची में एक सक्रिय एंट्री के रूप में (उसके पोर्ट और एक्शन आइकन के साथ), और अगर आप बाद में दूसरा इंस्टेंस शुरू करना चाहें तो अभी भी नीचे कैटलॉग में।

WHAT	Containers → Available containers → कैटलॉग में किसी चीज़ पर GET → डाउनलोड होने के बाद RUN।
WHY	दो चरणों में बाँटना (पहले डाउनलोड, फिर शुरू) का मतलब है कि आप तुरंत चलाने की प्रतिबद्धता के बिना पहले से एक इमेज ला सकते हैं — धीमे कनेक्शन पर या जब आप सिर्फ़ यह देख रहे हों कि क्या उपलब्ध है, तब उपयोगी।
RESULT	एक चल रहा कंटेनर, मुख्य सूची में अपने पोर्ट मैपिंग के साथ दिखाई देता है, वहाँ से आगे प्रबंधित किया जा सकता है।
RISK	कम — चुनिंदा कैटलॉग की सेल्फ़-होस्टेड सेवाएँ डिफ़ॉल्ट रूप से rootless-अनुकूल और कम जोखिम वाली चुनी गई हैं। एक्सपोज़र वही है जो किसी भी वेब सेवा को चलाने पर होता है: वह जिस भी पोर्ट पर सुन रहा हो, वह आपकी Firewall सेटिंग्स के अनुसार पहुँच योग्य होता है।

8.4 दृश्य गाइड — कंटेनर सूची और उसकी कार्रवाइयाँ

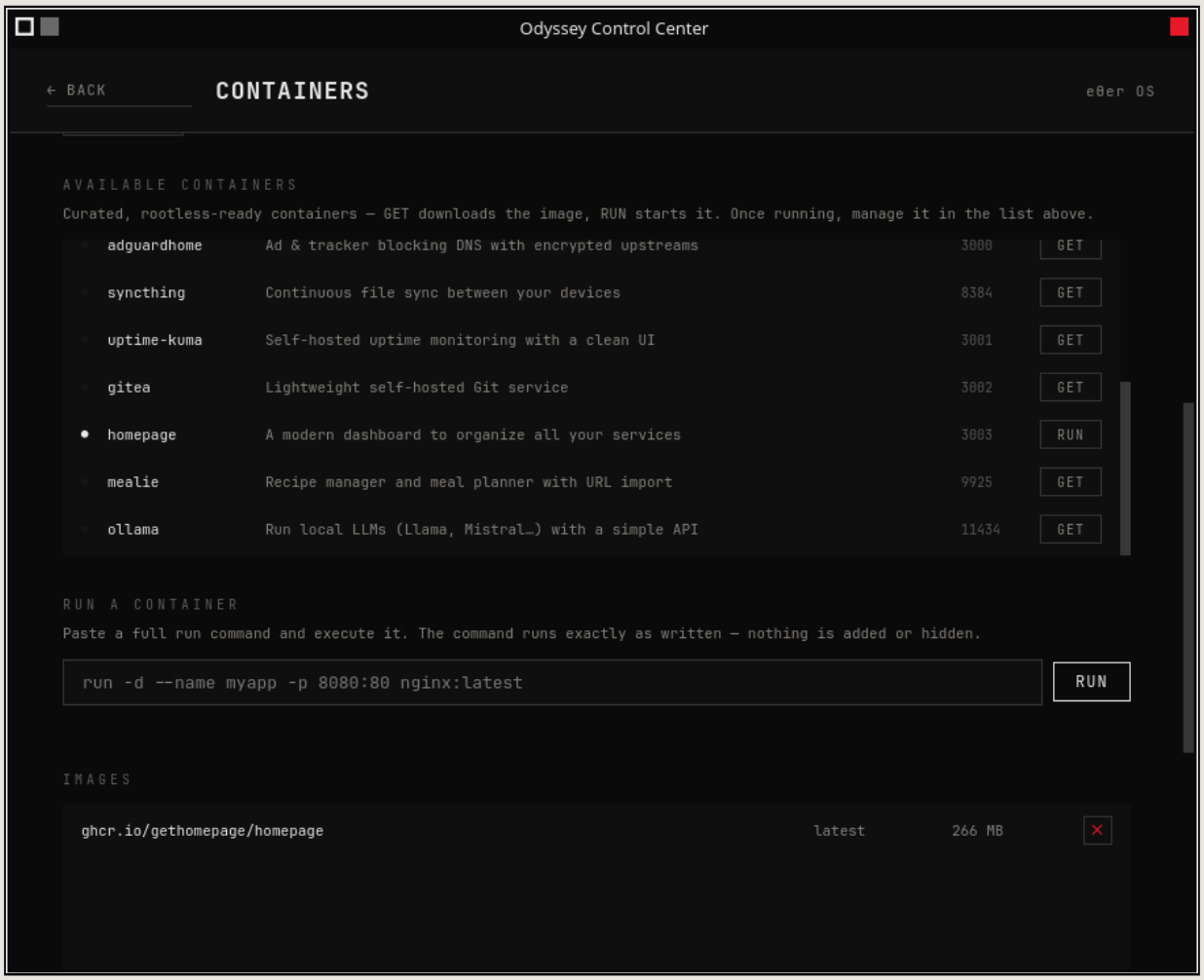

FIG · Containers सूची में homepage की लाइन, दाईं ओर उसके पाँच एक्शन आइकन दिखाई दे रहे हैं — नीचे उनकी व्याख्या है।

हर चल रहे कंटेनर की लाइन आइकन के एक सेट पर ख़त्म होती है — किसी भी आइकन पर माउस ले जाने से उसका लेबल दिखता है, लेकिन यहाँ है कि हर एक क्या करता है:

आइकन	कार्रवाई
खोलें (तीर)	सेवा को सीधे आपके ब्राउज़र में खोलता है — एक बार कुछ चलने लगे तो यह सबसे तेज़ रास्ता है
रोकें / शुरू करें (वर्ग / त्रिकोण)	कंटेनर को बिना हटाए रोक देता है, या उसे फिर से शुरू करता है
फिर से शुरू करें (गोलाकार तीर)	एक ही कार्रवाई में रोकता और शुरू करता है — "चल तो रहा है पर जवाब नहीं दे रहा" के लिए हल
logs (लाइनों वाला आइकन)	कंटेनर के अपने लॉग आउटपुट को खोलता है — अगर कुछ काम न कर रहा हो तो सबसे पहले यहीं देखना चाहिए
हटाएँ (×)	कंटेनर को पूरी तरह हटा देता है — इमेज ख़ुद डाउनलोड की हुई रहती है, अलग से Images में सूचीबद्ध

8.5 दृश्य गाइड — एक कंटेनर चलाना, और उसे खोलना

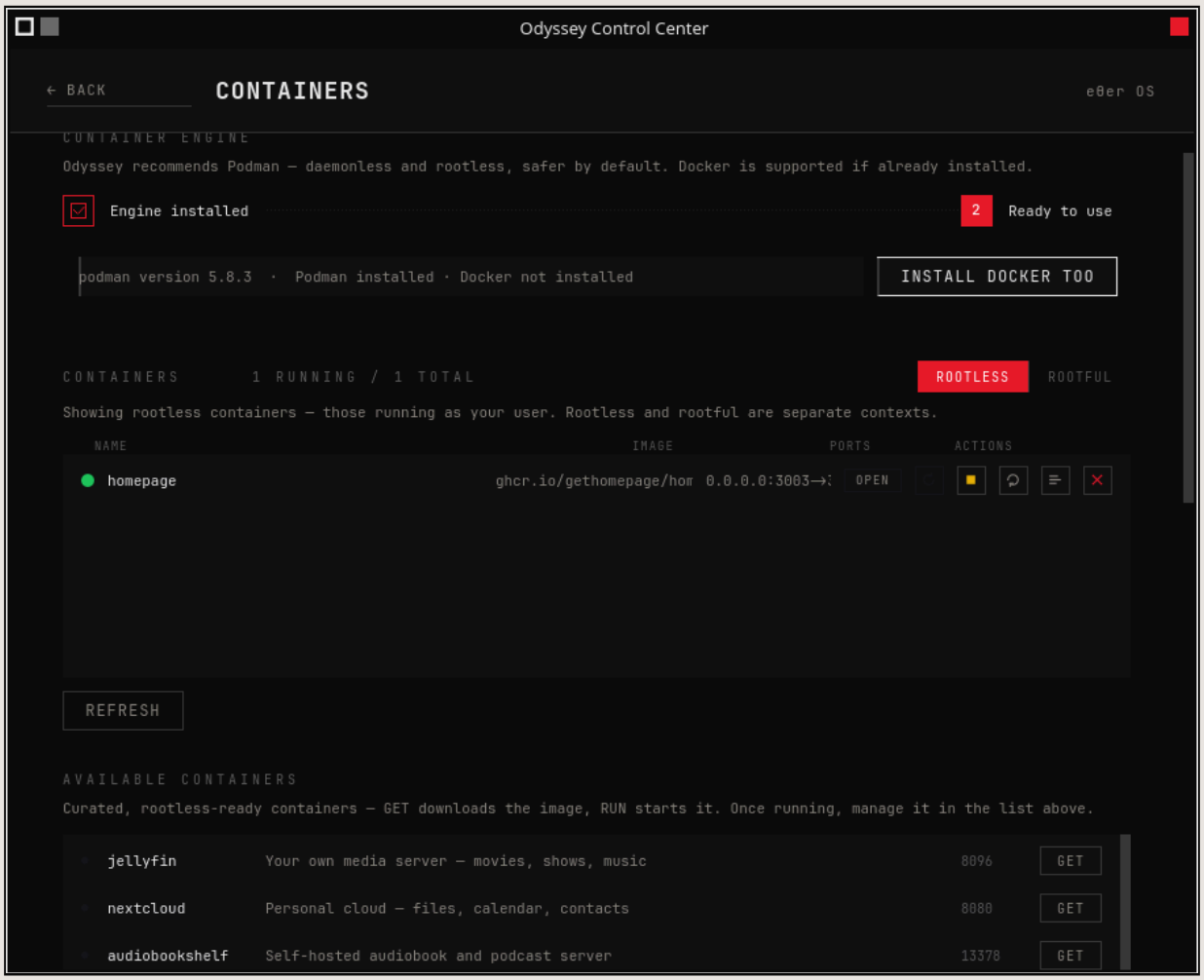

FIG · प्लेसहोल्डर टेक्स्ट के रूप में एक उदाहरण कमांड वाला "Run a container" फ़ील्ड, RUN बटन, और नीचे Images सेक्शन जिसमें ghcr.io/gethomepage/homepage · latest · 266 MB हटाने वाले बटन के साथ सूचीबद्ध है।

कैटलॉग में न होने वाली किसी भी चीज़ के लिए, **Run a container** एक पूरा `podman run` (या `docker run`) कमांड लेता है और उसे बिल्कुल वैसे ही चलाता है जैसे वह लिखा गया है — कुछ भी जोड़ा नहीं जाता, कुछ भी छिपाया नहीं जाता।

WHAT	Containers → "Run a container" → एक पूरा run कमांड चिपकाएँ (जैसे `run -d --name myapp -p 8080:80 nginx:latest`) → RUN।
WHY	चुनिंदा कैटलॉग सामान्य मामलों को कवर करता है; यह किसी ख़ास इमेज, ख़ास फ़्लैग, ख़ास पोर्ट के लिए बचाव का रास्ता है जो कैटलॉग नहीं देता।
RESULT	कंटेनर बिल्कुल वैसे ही शुरू होता है जैसा कमांड बताता है, और चलने के बाद मुख्य Containers सूची में शामिल हो जाता है।
RISK	कमांड ख़ुद जो भी करे — यह एक ईमानदार पास-थ्रू है, कोई सैंडबॉक्स नहीं। इसे उतनी ही सावधानी से इस्तेमाल करें जितनी टर्मिनल में वही कमांड टाइप करते समय बरतेंगे।

नीचे, **Images** हर डाउनलोड की गई इमेज को सूचीबद्ध करता है, चाहे अभी उससे कोई कंटेनर चल रहा हो या नहीं, **REFRESH** और **PRUNE UNUSED** के साथ, ताकि उन इमेज से जगह वापस ली जा सके जिन्हें अब कोई संदर्भित नहीं करता।

किसी चल रहे कंटेनर को खोलना

किसी चल रहे कंटेनर की लाइन पर OPEN पर क्लिक करना — ऊपर आइकन लाइन से वही खोलने वाली कार्रवाई — आपका डिफ़ॉल्ट ब्राउज़र सीधे उसके पते और पोर्ट पर ले जाता है, जैसे `homepage` के लिए `http://localhost:3003`। एक बार चलने लगे तो आपको ख़ुद पोर्ट नंबर याद रखने की ज़रूरत नहीं।

8.6 डिस्क इस्तेमाल

पैनल के बिल्कुल नीचे, तीन रीयल-टाइम कुल — कंटेनर, इमेज, कुल आकार — ताकि आपको हमेशा पता रहे कि कंटेनर डिस्क पर कितनी जगह ले रहे हैं, बिना टर्मिनल खोले।

```
▸ ADVANCED

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

8.7 Advanced

कमांड-लाइन के बराबर विकल्प

```
# जिसे पैनल "GET" कहता है
$ podman pull ghcr.io/gethomepage/homepage
# जिसे पैनल किसी कैटलॉग एंट्री पर "RUN" कहता है
$ podman run -d -name homepage -p 3003:3000 ghcr.io/gethomepage/homepage
# चल रहे कंटेनर सूचीबद्ध करता है — rootless संदर्भ
$ podman ps
# वही, rootful
$ sudo podman ps
# logs, रोकना, फिर से शुरू करना, हटाना — आइकन लाइन, हाथ से
$ podman logs -f homepage
$ podman stop homepage
$ podman restart homepage
$ podman rm homepage
# इमेज और डिस्क इस्तेमाल
$ podman images
$ podman system df
$ podman image prune
```

ROOTLESS बनाम ROOTFUL, अंदर से

Rootless Podman कंटेनरों को आपके अपने यूज़र के तौर पर, एक यूज़र नेमस्पेस के भीतर चलाता है — कंटेनर का "root" होस्ट पर बिना विशेषाधिकार वाले UID की एक रेंज से मैप होता है, इसलिए भले ही कंटेनर प्रोसेस पूरी तरह से समझौता कर लिया जाए, वह फिर भी आपके अपने अकाउंट की अनुमतियों तक सीमित रहता है। Rootful Podman (`sudo podman ...`) पारंपरिक तरीक़े से काम करता है, कंटेनर का root होस्ट के असली root से मैप होता है। पैनल का स्विच असल में सिर्फ़ सामान्य `podman` या `sudo podman` चलाने के बीच बदलता है — हर संदर्भ का अपना, सच में अलग स्थिति डायरेक्टरी है, अलग कंटेनर सूचियाँ हैं, डिफ़ॉल्ट रूप से दोनों संदर्भों के बीच कुछ भी साझा नहीं होता।

```
# हर संदर्भ अपनी स्थिति कहाँ रखता है
$ podman info -format '{{.Store.GraphRoot}}'
# rootless:  /.local/share/containers/storage
# rootful: /var/lib/containers/storage
```

कुछ कैटलॉग एंट्रीज़ को ख़ास पोर्ट खोलने की ज़रूरत क्यों पड़ती है

किसी कंटेनर की पोर्ट मैपिंग (homepage उदाहरण में `3003:3000`) डिफ़ॉल्ट रूप से सेवा को **सिर्फ़ इसी मशीन से** पहुँच योग्य बनाती है — जब तक अलग से न बताया जाए, Podman इसे `localhost` से बाँधता है। अपने नेटवर्क में किसी दूसरी डिवाइस से इसे पहुँचने के लिए इसके अलावा Firewall पैनल (अध्याय 4) में उस पोर्ट का खुला होना ज़रूरी है — दोनों सिस्टम स्वतंत्र परतें हैं, और मैप किए गए पोर्ट वाला चल रहा कंटेनर सिर्फ़ इसलिए इंटरनेट या LAN से अपने आप पहुँच योग्य नहीं हो जाता क्योंकि Podman उस पर सुन रहा है।

CHAPTER 9

Gaming Mode

ऊपर एक मुख्य स्विच, रीयल-टाइम स्टेटस की एक लाइन, और चार टैब: **PACKAGES**, **GAMEMODE**, **MANGOHUD**, **PROTON**।

9.1 दृश्य गाइड — मुख्य स्विच

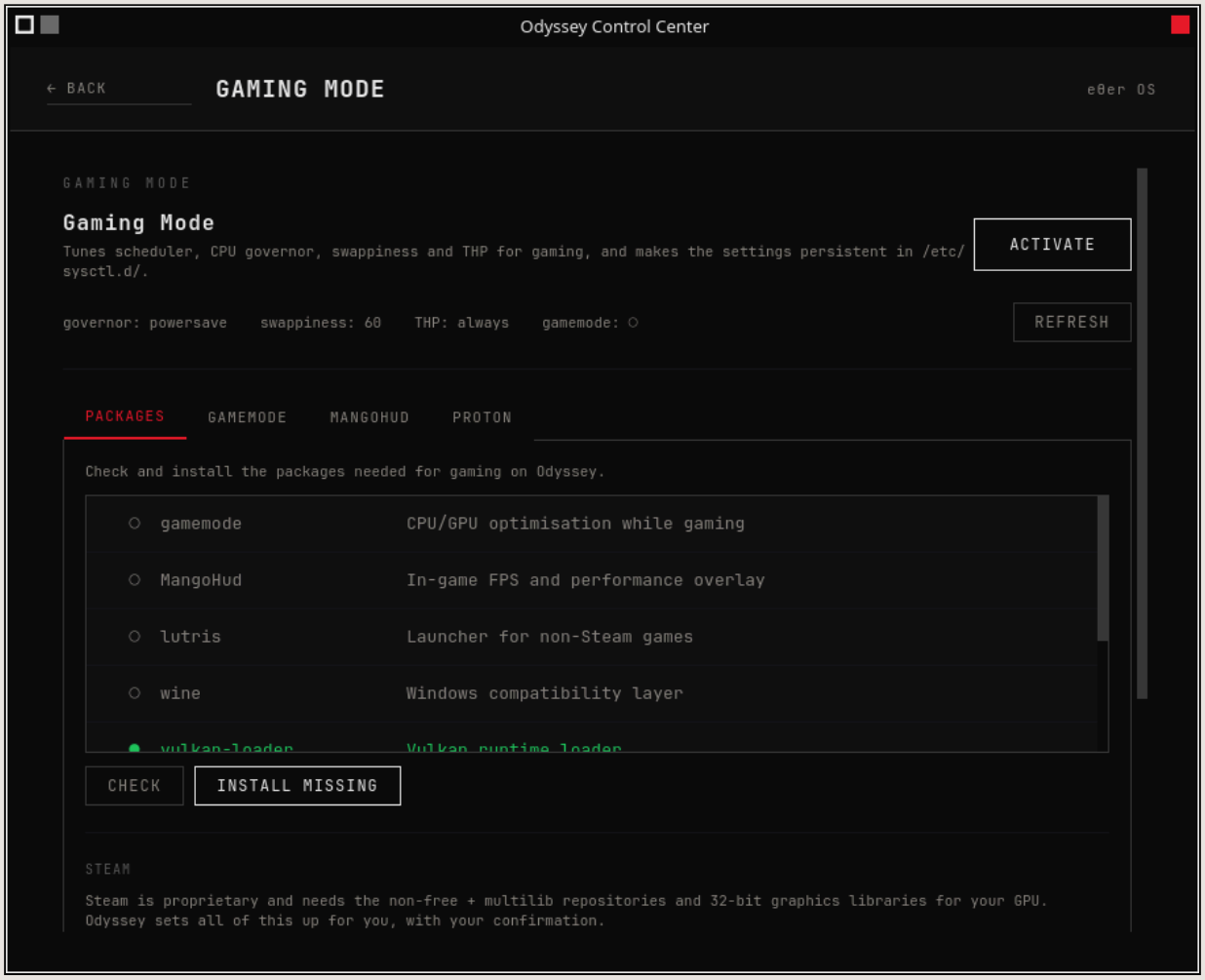

FIG · पैनल का ऊपरी सेक्शन: “Gaming Mode — Tunes scheduler, CPU governor, swappiness and THP for gaming, and makes the settings persistent in /etc/sysctl.d/”, एक ACTIVATE बटन, एक REFRESH बटन, और लाइव स्टेटस लाइन: governor powersave, swappiness 60, THP always, gamemode off।

यह स्टेटस लाइन वही लाइव मान दिखाती है जो Kernel और Memory पैनल दिखाते हैं — governor, swappiness, THP स्थिति, और क्या `gamemode` सेवा चल रही है — ताकि आप एक नज़र में पुष्टि कर सकें कि क्या चालू करना असल में असरदार हुआ, बिना उन दोनों पैनल को अलग-अलग देखे।

WHAT	Gaming Mode → ACTIVATE।
WHY	गेमिंग के लिए उपयुक्त governor, swappiness और THP मान हाथ से सेट करने के लिए Kernel और Memory में जाना, ठीक उसी तरह का बहु-चरणीय काम है जिसे एक समर्पित मोड को ख़त्म करना चाहिए।
RESULT	Scheduler, governor, swappiness और THP सभी गेमिंग के लिए उपयुक्त मानों पर चले जाते हैं, तुरंत लागू और स्थायी हो जाते हैं — स्टेटस लाइन बदलाव को दिखाने के लिए अपडेट होती है।
RISK	कम, और उलटा जा सकता है — वही मान जो आप अन्यथा हाथ से Kernel/Memory में सेट करते, बस एक क्लिक में इकट्ठे किए गए।

9.2 दृश्य गाइड — Packages और Steam

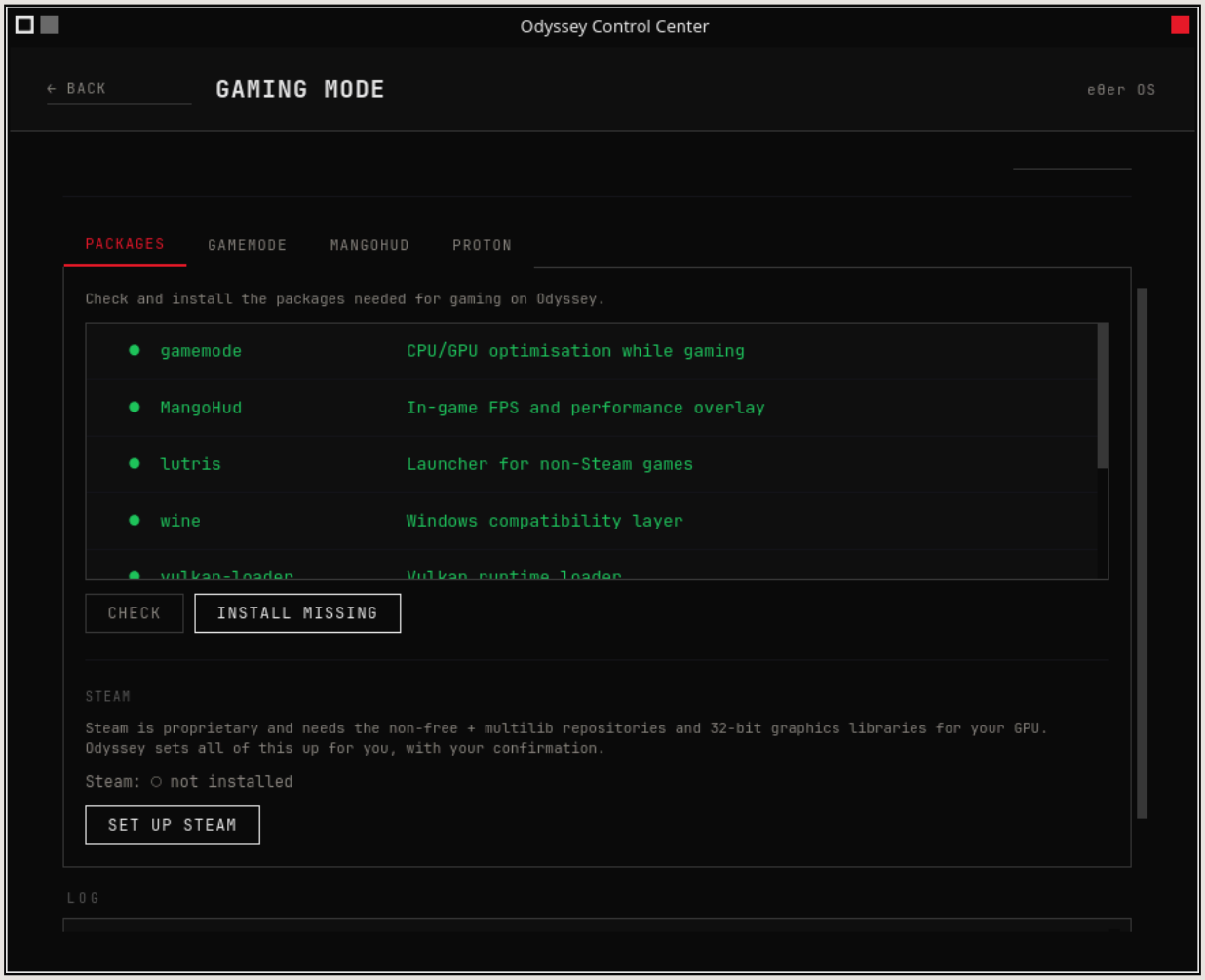

FIG · PACKAGES टैब: gamemode, MangoHud, lutris, wine, vulkan-loader सब भरी हुई हरी बिंदी (इंस्टॉल) के साथ; ऊपर CHECK और INSTALL MISSING; नीचे Steam सेक्शन “Steam: ● installed” के साथ और REINSTALL STEAM बटन।

गेमिंग के लिए आम तौर पर ज़रूरी पैकेजों की एक जाँच-सूची — CHECK यह देखता है कि क्या कमी है, INSTALL MISSING एक ही क्रिया में उन कमियों को पूरा करता है। एक बार सब कुछ भरी हुई हरी बिंदी दिखाए, जैसा स्क्रीनशॉट में है, आप पूरी तरह से तैयार हैं।

Steam की जाँच-सूची के नीचे उसकी अपनी समर्पित प्रक्रिया है, क्योंकि इसे एक साधारण पैकेज इंस्टॉलेशन से ज़्यादा की ज़रूरत है।

1 **SET UP STEAM**

क्लिक करें — यह अभी कुछ भी इंस्टॉल नहीं करता, पहले एक जानकारी देने वाला पुष्टिकरण डायलॉग खोलता है।

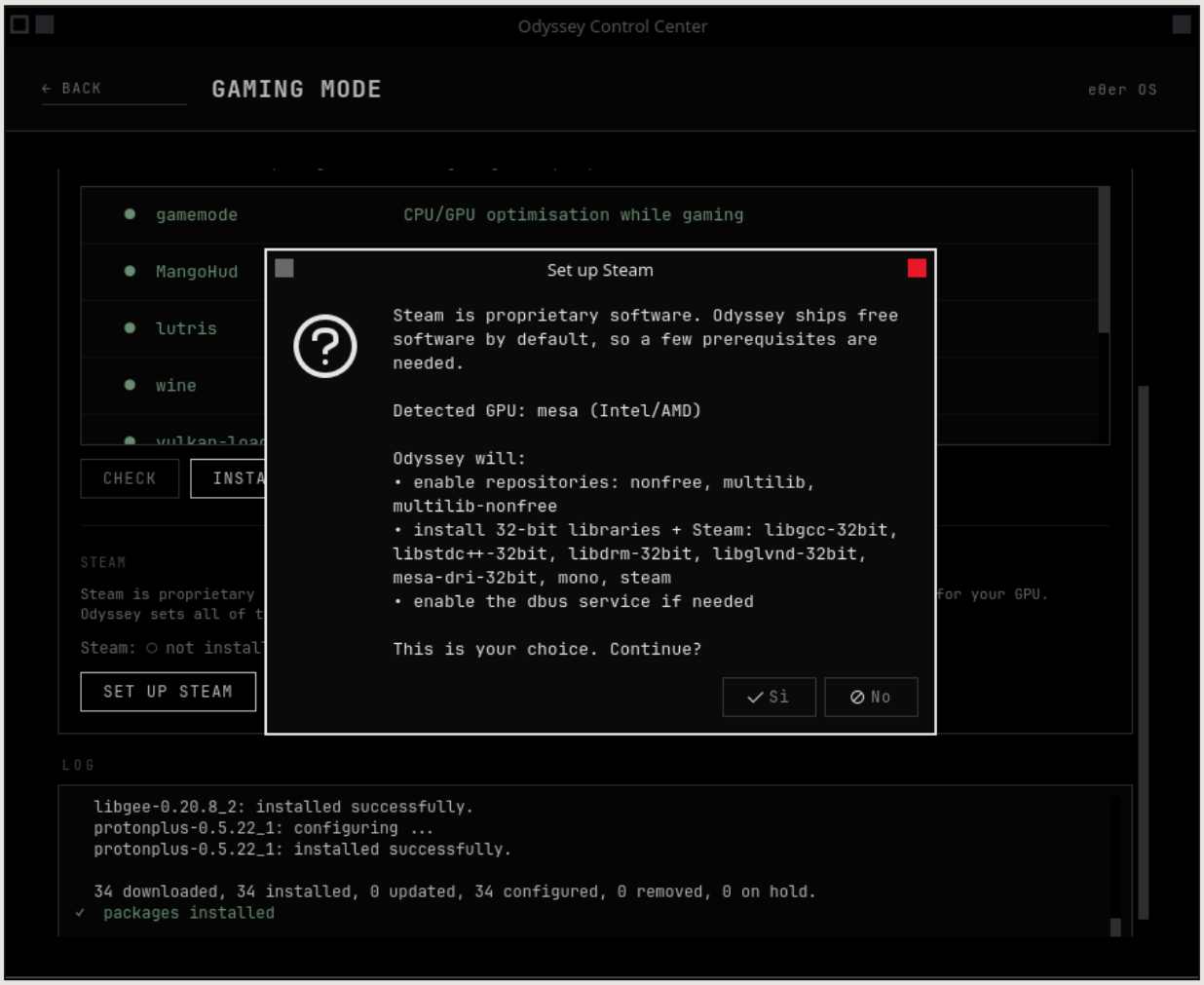

FIG · पुष्टिकरण डायलॉग: “Steam is proprietary software. Odyssey ships free software by default, so a few prerequisites are needed. Detected GPU: mesa (Intel/AMD)।” बुलेट सूची जिसमें बताया गया है कि क्या-क्या चालू और इंस्टॉल होगा, आख़िर में “This is your choice. Continue?” के साथ Sí / No बटन।

आपके सहमत होने से पहले यह डायलॉग बिल्कुल बताता है कि क्या होने वाला है: कौन-सी रिपॉज़िटरी चालू होंगी (`nonfree`, `multilib`, `multilib-nonfree`), आपके **पहचाने गए** GPU के मुताबिक़ कौन-सी 32-बिट लाइब्रेरी इंस्टॉल होंगी, और अगर dbus सेवा पहले से चालू नहीं है तो वह चालू हो जाएगी। पुष्टि के पीछे कुछ भी छिपाया नहीं गया है।

2 **Sí**

इसकी पुष्टि करता है। लॉग असली इंस्टॉलेशन भेजता है — पैकेज डाउनलोड, कॉन्फ़िगरेशन, पूरा होना।

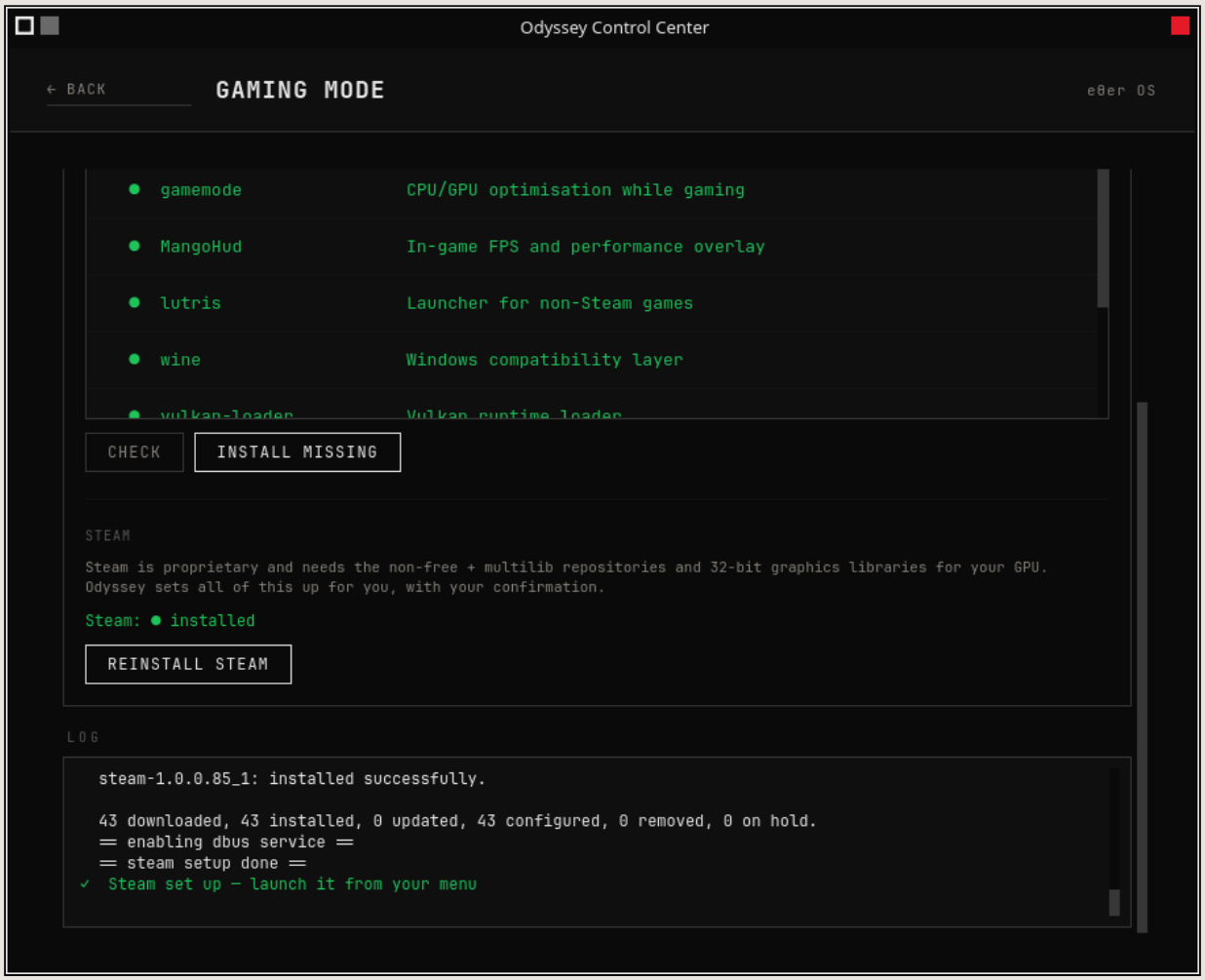

FIG · सेटअप के बाद PACKAGES टैब: सभी पैकेज हरे, Steam "● installed" दिखाता है REINSTALL STEAM के साथ, और लॉग "Steam set up — launch it from your menu." पर ख़त्म होता है।

WHAT	Gaming Mode → PACKAGES टैब → SET UP STEAM → डायलॉग की समीक्षा करें → Sí।
WHY	Steam मालिकाना है और इसे non-free/multilib रिपॉज़िटरी के साथ-साथ GPU-विशेष 32-बिट लाइब्रेरी चाहिए जो Void डिफ़ॉल्ट रूप से चालू नहीं करता — Void पर आधारित सिस्टम के साथ शुरुआत करने वालों के लिए इसे हाथ से सही करना एक आम रुकावट है।
RESULT	ज़रूरी रिपॉज़िटरी, Steam ख़ुद, और आपके ख़ास GPU के मुताबिक़ 32-बिट ग्राफ़िक्स लाइब्रेरी, सब इंस्टॉल — इसके बाद अपने ऐप्लिकेशन मेन्यू से Steam लॉन्च करें।
RISK	कम — यह सिर्फ़ रिपॉज़िटरी चालू करता है और पैकेज जोड़ता है; मौजूदा किसी भी चीज़ को हटाया या फिर से कॉन्फ़िगर नहीं किया जाता। पुष्टि करने से पहले डायलॉग दिखाता है कि क्या-क्या होगा, इसकी पूरी सूची।

9.3 GameMode और MangoHud

GameMode (अपना टैब) Feral Interactive का प्रति-गेम परफ़ॉर्मेंस डेमन है — इसके ज़रिए लॉन्च किया गया कोई गेम सिर्फ़ अपनी प्रोसेस के लिए एक अस्थायी परफ़ॉर्मेंस बढ़ोतरी (CPU governor, I/O प्राथमिकता) माँगता है, जो गेम बंद होने पर अपने आप वापस लौट जाती है। **MangoHud** (अपना टैब) एक स्क्रीन ओवरले है जो गेमिंग के दौरान FPS, फ़्रेम टाइम, CPU/GPU लोड और तापमान दिखाता है। दोनों टैब इंस्टॉल/जाँच के नियंत्रण देते हैं, और MangoHud के लिए, ओवरले के दिखने के तरीक़े का तेज़ कॉन्फ़िगरेशन भी।

9.4 Proton

Steam के ज़रिए Windows गेम चलाने के लिए संगतता परत का प्रबंधन — जो इंस्टॉल और उपलब्ध है उसकी जाँच करना, और जब किसी ख़ास गेम को ऐसे Proton वर्ज़न की ज़रूरत हो जिसे Valve डिफ़ॉल्ट रूप से नहीं बाँटता, तो Proton-GE जैसे कम्युनिटी बिल्ड सहित अतिरिक्त Proton वर्ज़न इंस्टॉल करना।

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

9.5 Advanced

ACTIVATE क्या लिखता है

Kernel और Memory पैनल के MAKE PERSISTENT ऐक्शन जैसी ही अंतर्निहित फ़ाइलें — Gaming Mode एक बंडल है, कोई अलग तंत्र नहीं:

```
# स्थायी बनाए गए sysctl मान
$ cat /etc/sysctl.d/99-odyssey.conf
# CPU governor — यह sysctl नहीं है, sysfs पर आधारित है, इस तंत्र से
# स्थायी नहीं होता; अगर आप इस पर निर्भर हैं तो बूट के बाद फिर से लागू करें
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

SET UP STEAM असल में क्या चलाता है

पुष्टिकरण डायलॉग ख़ुद लगभग एक पूरा तकनीकी खुलासा है, लेकिन ठोस रूप से, एक Mesa (Intel/AMD) सिस्टम पर:

```
$ 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 सिस्टम mesa-dri-32bit की जगह अतिरिक्त nvidia-libs-32bit डाउनलोड करते हैं —
# पैनल आपके GPU बनाने वाले की पहचान करता है और उसी के मुताबिक़ पैकेज सूची समायोजित करता है
```

GAMEMODE और MANGOHUD हाथ से

```
# GameMode चालू करके कोई भी गेम लॉन्च करें
$ gamemoderun ./game_binary
# या, किसी Steam गेम के लिए, लॉन्च ऑप्शन:
$ gamemoderun %command%
# MangoHud ओवरले, लॉन्च ऑप्शन:
$ mangohud %command%
# ओवरले क्या दिखाता है उसे समायोजित करने के लिए MangoHud की अपनी कॉन्फ़िगरेशन फ़ाइल
$ $EDITOR  /.config/MangoHud/MangoHud.conf
```

PROTON-GE का कोई ख़ास वर्ज़न इंस्टॉल करना

PROTON टैब `protonplus` या इसके बराबर किसी वर्ज़न मैनेजर को रैप करता है — हाथ से करने का बराबर तरीक़ा GitHub पर `GloriousEggroll/proton-ge-custom` से कोई रिलीज़ डाउनलोड करना और उसे Steam की संगतता टूल डायरेक्टरी में एक्सट्रैक्ट करना है:

```
$ mkdir -p  /.steam/root/compatibilitytools.d
$ tar -xf GE-Proton-VERSION.tar.gz -C  /.steam/root/compatibilitytools.d
# फिर Steam में हर गेम के लिए चुनें: Properties → Compatibility
```

CHAPTER 10

Telemetry

नाम के बावजूद, यह पैनल कोई ऐसी सेटिंग नहीं है जिसे आप कॉन्फ़िगर करें — यह एक रीयल-टाइम मॉनिटर है। Odyssey आपके बारे में कुछ भी इकट्ठा नहीं करता, कुछ भी नहीं भेजता, इसलिए यहाँ बंद करने के लिए कुछ भी नहीं है। इसके बजाय यह पैनल जो करता है वह है आपको रीयल-टाइम में हर वह बाहर जाने वाला कनेक्शन दिखाना जो आपका **अपना** सिस्टम बना रहा है — ताकि "मेरी मशीन अभी किससे बात कर रही है" ऐसा सवाल बन जाए जिसका जवाब आप देखकर दे सकें, किसी दावे पर भरोसा करके नहीं। यह ख़ुद को **Reverse Telemetry** इसी वजह से कहता है: यह निगरानी को उलट देता है, आपकी ओर से इसे सिस्टम पर ही केंद्रित करते हुए।

10.1 दृश्य गाइड

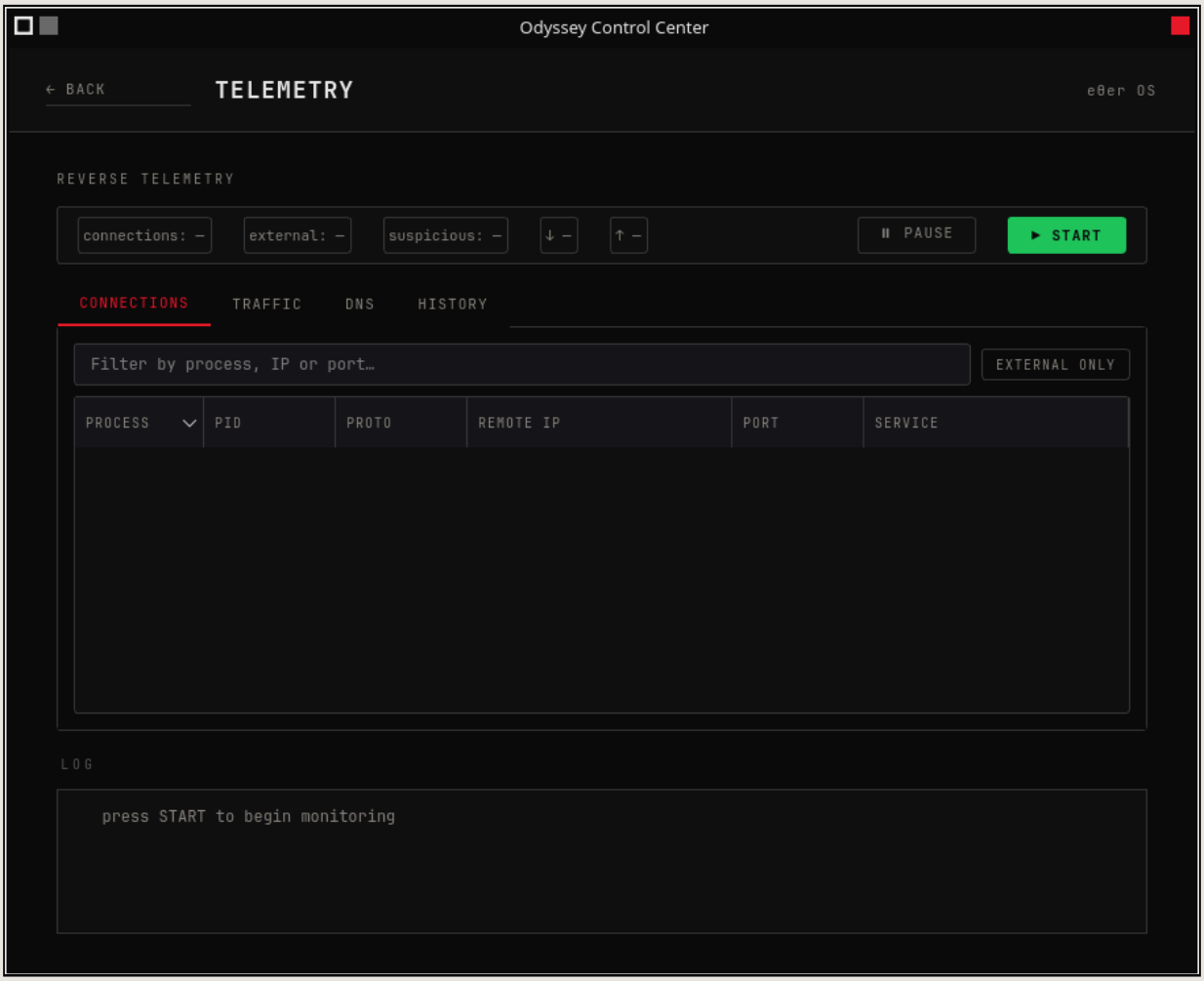

FIG · शुरू होने से पहले: connections/external/suspicious सभी "–" दिखा रहे हैं, एक हरा START बटन, और लॉग कहता है "press START to begin monitoring"। चार टैब — CONNECTIONS, TRAFFIC, DNS, HISTORY — एक भरी न हुई टेबल के ऊपर ख़ाली रहते हैं।

1 **START**

रीयल-टाइम मॉनिटरिंग शुरू करता है — स्टेटस बार और सक्रिय टैब तुरंत भरने लगते हैं।

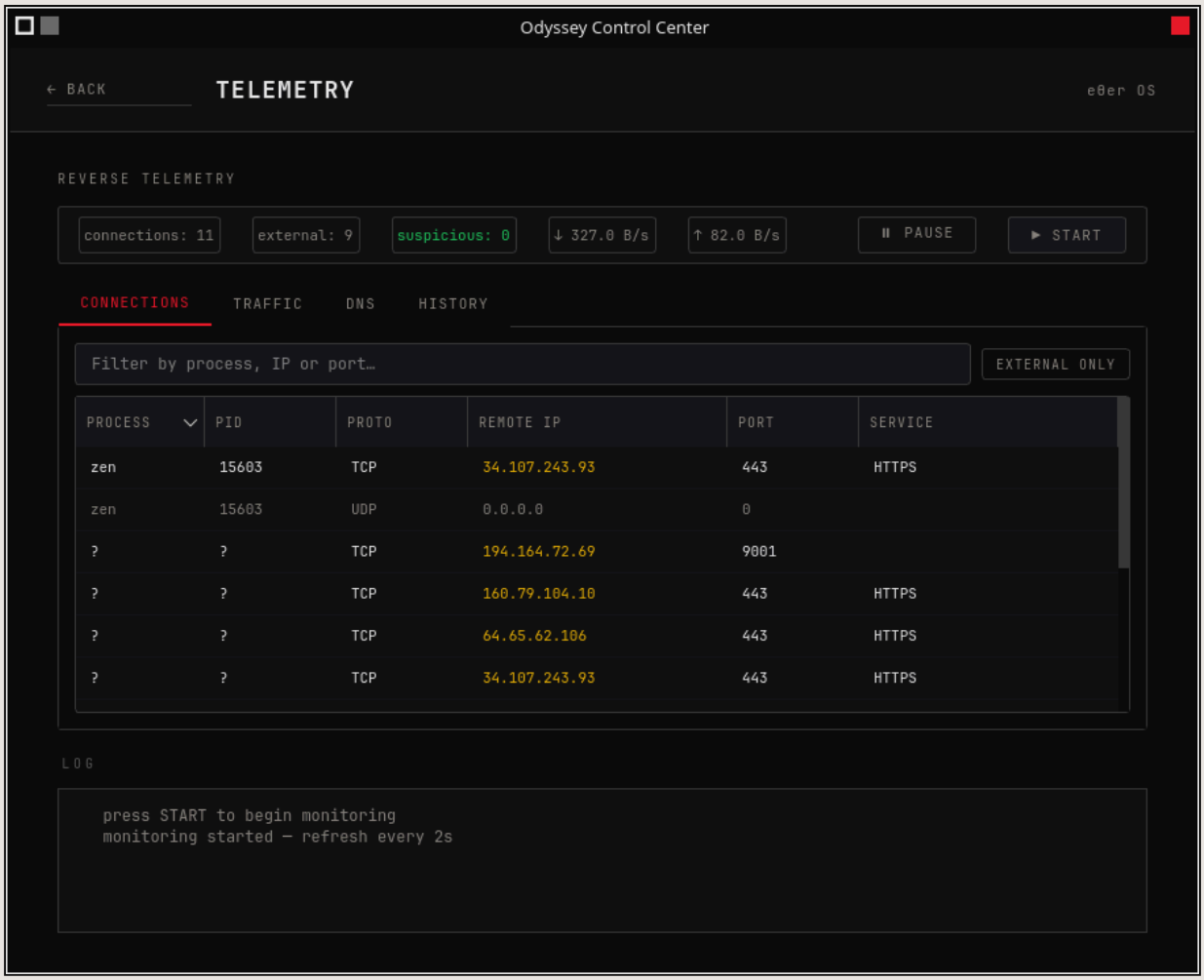

FIG · शुरू होने के बाद: "connections: 11 · external: 9 · suspicious: 0 · ↓327.0 B/s · ↑82.0 B/s", PAUSE अब उपलब्ध है, और CONNECTIONS टेबल असली लाइनों से भरी है: प्रोसेस "zen" TCP पर 34.107.243.93:443 (HTTPS) से, कई लाइनें प्रोसेस/PID के लिए "?" चिह्नित हैं दूसरे पतों से कनेक्शन के साथ।

ऊपर का स्टेटस बार रीयल-टाइम कुल देता है — कनेक्शन, बाहरी (स्थानीय के मुक़ाबले), संदिग्ध, मौजूदा डाउनलोड/अपलोड की रफ़्तार — और **PAUSE** पहले से मिले डेटा को खोए बिना नज़ारे को रोक देता है, **START** उसे फिर से शुरू करता है।

चार टैब

टैब	क्या दिखाता है
CONNECTIONS	हर सक्रिय कनेक्शन के लिए एक लाइन: प्रोसेस, PID, प्रोटोकॉल, दूरस्थ पता, पोर्ट, और उस पोर्ट के लिए जानी जाने वाली सेवा (443 → HTTPS, 22 → SSH)। किसी भी कॉलम से क्रमबद्ध किया जा सकता है।
TRAFFIC	हर इंटरफ़ेस के लिए थ्रूपुट — मौजूदा डाउनलोड/अपलोड रफ़्तार, साथ ही मॉनिटरिंग शुरू होने के बाद से जमा हुए कुल।
DNS	डोमेन नाम रिज़ॉल्यूशन का एक रीयल-टाइम प्रवाह, जैसे-जैसे होते हैं — अक्सर सिर्फ़ IP से ज़्यादा साफ़, क्योंकि होस्टनेम आपको **क्या** बताता है, सिर्फ़ **कहाँ** नहीं।
HISTORY	सत्र ने जो कुछ भी दर्ज किया है, वह सब स्क्रॉल होने वाले लॉग के रूप में — किसी छोटी अवधि की चीज़ को पकड़ने में उपयोगी जो आपके देखने तक सक्रिय कनेक्शन नहीं रह जाती।

जिन लाइनों में प्रोसेस ? दिखाता है (स्क्रीनशॉट में दिखाई देता है) उनका मतलब है कि कनेक्शन दर्ज हो गया लेकिन उस पल स्रोत प्रोसेस को PID से पहचाना नहीं जा सका — बहुत कम समय के कनेक्शन में यह आम है। टेबल के ऊपर एक **फ़िल्टर** फ़ील्ड प्रोसेस, IP या पोर्ट के हिसाब से सीमित करता है, और **EXTERNAL ONLY** सिर्फ़ स्थानीय ट्रैफ़िक को छिपा देता है।

WHAT	Telemetry → START → CONNECTIONS देखें, या किसी ख़ास चीज़ की जाँच के लिए PROCESS या REMOTE IP से क्रमबद्ध करें।
WHY	यह इस सवाल का सीधा, ठोस जवाब है कि “क्या इस मशीन पर कुछ ऐसा है जो नेटवर्क तक पहुँच रहा है जिसे नहीं पहुँचना चाहिए” — सीधे कर्नेल की अपनी कनेक्शन टेबल से पढ़ा गया, किसी विक्रेता के अपने सॉफ़्टवेयर के बारे में दावे से नहीं।
RESULT	हर बाहर जाने वाले कनेक्शन की एक लाइव, सटीक तस्वीर, जो तब तक लगातार अपडेट होती रहती है जब तक यह चल रहा है और रुका नहीं है।
RISK	कोई नहीं — पूरी तरह केवल-पढ़ने वाला।

जिन प्रोसेस के पास ख़ुद नेटवर्क तक पहुँचने की कोई सामान्य वजह नहीं होती — कोई खुला shell, कोई स्वतंत्र रूप से चल रहा स्क्रिप्ट इंटरप्रेटर — उनसे आने वाले कनेक्शन स्टेटस बार में अपने आप “suspicious” के तौर पर चिह्नित हो जाते हैं, ताकि कोई असामान्य चीज़ आपके ब्राउज़र और सिस्टम सेवाओं के सामान्य ट्रैफ़िक में दबने के बजाय उभर कर सामने आए।

▸ ADVANCED

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

10.2 Advanced

डेटा कहाँ से आता है

पैनल वही कर्नेल-स्तर की कनेक्शन टेबल पढ़ता है जो कोई भी मानक टूल पढ़ता, मॉनिटरिंग सक्रिय रहते हुए एक छोटे अंतराल पर अपडेट होती है — इसका अपना कोई ख़ास ड्राइवर या कर्नेल मॉड्यूल नहीं है। कमांड-लाइन के बराबर विकल्प:

```
# हर खुला सॉकेट, उसे रखने वाली प्रोसेस के साथ — CONNECTIONS जैसे ही फ़ील्ड
$ sudo ss -tupn
# हर इंटरफ़ेस का थ्रूपुट — TRAFFIC जैसी ही संख्याएँ
$ ip -s link
# अगर आप इसे पैनल के बाहर देखना चाहें, तो एक बार का लाइव व्यू
$ sudo nethogs
```

DNS रिज़ॉल्यूशन, हाथ से

```
# बाहर जाने वाली DNS क्वेरीज़ का लाइव कैप्चर (पैकेट कैप्चर टूल चाहिए)
$ sudo tcpdump -i any -n port 53
```

कुछ लाइनें PROCESS को “?” क्यों दिखाती हैं

किसी लाइव कनेक्शन को उसे रखने वाली प्रोसेस से मिलाने के लिए कैप्चर के उस पल हर संभावित प्रोसेस का `/proc/PID/fd` पढ़ना पड़ता है — अगर कनेक्शन पैनल के अपडेट अंतराल से भी तेज़ खुलता और बंद हो जाता है, या रखने वाली प्रोसेस पहले ही ख़त्म हो चुकी है, तो पैनल के पास यह मिलान करने के लिए वह खिड़की नहीं होती और वह अंदाज़ा लगाने के बजाय `?` दिखाता है। यह एक समय संबंधी सीमा है, इस बात का संकेत नहीं कि कुछ छिपाया जा रहा है।

CHAPTER 11

About

मशीन की एक ही, विस्तृत डायग्नोस्टिक शीट, जो हर बार जब आप टैब खोलते हैं तब अपडेट होती है, ताकि यह हमेशा वही दिखाए जो दूसरे पैनल में बदला है, बिना Control Center को फिर से खोलने की ज़रूरत के।

11.1 दृश्य गाइड

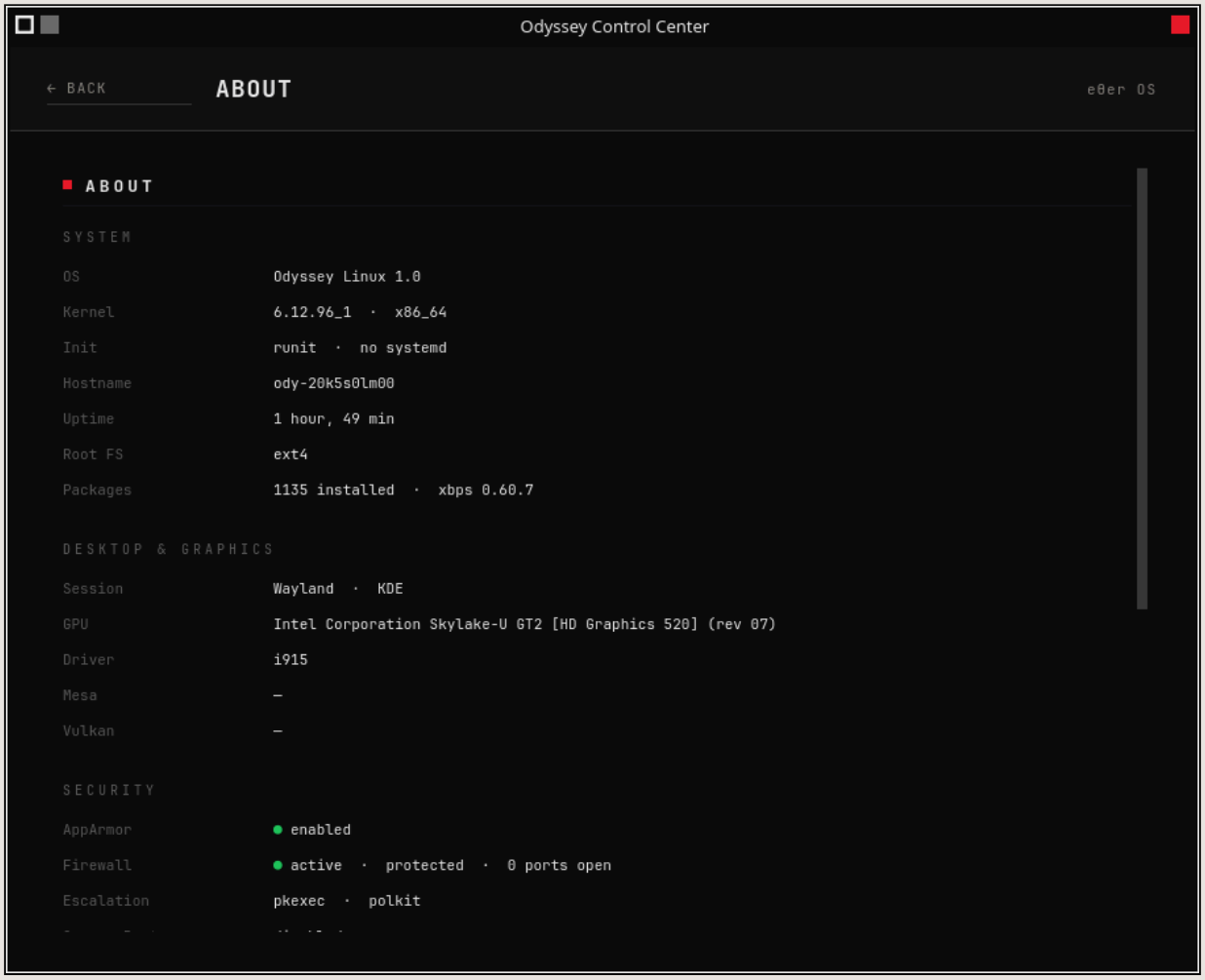

FIG · पैनल बिल्कुल ऊपर तक स्क्रॉल किया हुआ: SYSTEM सेक्शन OS “Odyssey Linux 1.0”, Kernel 6.12.96_1 · x86_64, Init runit · no systemd, Hostname, Uptime, Root FS ext4, Packages “1135 installed · xbps 0.60.7” दिखा रहा है; नीचे DESKTOP & GRAPHICS है Session, GPU, Driver, Mesa, Vulkan के साथ; SECURITY AppArmor “● enabled” और Firewall “● active · protected · 0 ports open” दिखाना शुरू कर रहा है।

क्रम में तीन सेक्शन: **SYSTEM** (कर्नेल, init, होस्टनेम, अपटाइम, फ़ाइल सिस्टम, पैकेज गिनती), **DESKTOP & GRAPHICS** (सेशन का प्रकार, GPU, ड्राइवर, Mesa/Vulkan सपोर्ट), और उससे नीचे **SECURITY** (AppArmor और Firewall की स्थिति रंगीन संकेतकों के साथ, सिर्फ़ टेक्स्ट नहीं — चालू/सक्रिय के लिए हरा) और **HARDWARE** (मदरबोर्ड, ऑडियो)।

WHAT	About → तीनों सेक्शन में स्क्रॉल करके किसी भी मान की जाँच करें — कर्नेल का वर्ज़न, इस्तेमाल हो रहा ड्राइवर, क्या AppArmor और फ़ायरवॉल वाक़ई सक्रिय हैं।
WHY	यह पाँच अलग-अलग पैनल खोले बिना, एक ही जगह पूरे सिस्टम की स्थिति की पुष्टि करने का सबसे तेज़ तरीक़ा है।
RESULT	एक पूरा, अभी का स्नैपशॉट — हर बार टैब खोलने पर अपडेट होता है, इसलिए यह कभी भी किसी पुराने सत्र की बासी जानकारी नहीं दिखाता।
RISK	कोई नहीं — पूरी तरह केवल-पढ़ने वाला।

11.2 Copy report

पैनल के ऊपर सिर्फ़ एक बटन, **COPY REPORT**, पूरी शीट को सादे टेक्स्ट के रूप में क्लिपबोर्ड में रख देता है — हर सेक्शन, हर मान — सीधे किसी फ़ोरम पोस्ट या bug tracker की एंट्री में चिपकाने के लिए तैयार।

WHAT	About → COPY REPORT → अपनी बग रिपोर्ट या फ़ोरम पोस्ट में चिपकाएँ।
WHY	बग रिपोर्ट उतनी ही उपयोगी होती हैं जितने साथ लगे सिस्टम विवरण — कर्नेल का वर्ज़न, ड्राइवर, AppArmor की स्थिति और दर्जन भर और फ़ील्ड हाथ से टाइप करना ठीक वही घर्षण है जिसकी वजह से बहुत से लोग यह चरण छोड़ देते हैं।
RESULT	क्लिपबोर्ड में एक पूरी, सही तरीक़े से फ़ॉर्मेट की गई सिस्टम रिपोर्ट, चिपकाने के लिए तैयार — वही जानकारी जो एक मेंटेनर को अन्यथा आपसे फ़ील्ड-दर-फ़ील्ड पूछनी पड़ती।
RISK	आपके सिस्टम के लिए कोई नहीं। रिपोर्ट डायग्नोस्टिक जानकारी है (वर्ज़न, स्थितियाँ, हार्डवेयर मॉडल) — इसमें व्यक्तिगत फ़ाइलें या क्रेडेंशियल्स शामिल नहीं हैं — लेकिन किसी भी डायग्नोस्टिक डंप की तरह, अगर आप निश्चित नहीं हैं कि इसमें क्या है, तो इसे कहीं सार्वजनिक रूप से पोस्ट करने से पहले फिर से पढ़ लें।

▸ ADVANCED

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

11.3 Advanced

हर फ़ील्ड कहाँ से आती है

इस पैनल में कुछ भी गढ़ा हुआ या इंस्टॉलेशन के समय से कैश किया हुआ नहीं है — हर मान लाइव पूछा जाता है, हर बार जब टैब खुलता है:

```
$ 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
```

Security सेक्शन विशेष रूप से:

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

पहले ABOUT क्यों जाँचना फ़ायदेमंद है

क्योंकि यह हर बार जाने पर सिस्टम को फिर से पूछता है, About किसी दूसरे पैनल में किए गए बदलाव के वाक़ई असर करने की पुष्टि करने का सबसे तेज़ तरीक़ा है — Firewall में कोई फ़ायरवॉल मोड लागू करें, फिर ख़ुद Firewall को फिर से खोलने के बजाय About का Security सेक्शन जाँचें। यह बाक़ी Control Center के लिए “क्या यह वाक़ई काम कर गया?” वाली क्रॉस-चेक बनने के लिए बनाया गया है।

CHAPTER 12

NVIDIA driver

एकीकरण बाक़ी है

यह अध्याय एक प्लेसहोल्डर है। NVIDIA मॉड्यूल Control Center में सिर्फ़ उन मशीनों पर दिखाई देता है जहाँ NVIDIA हार्डवेयर मिला हो, और इसके लिए स्क्रीनशॉट सिर्फ़ ऐसी ही मशीन पर लिए जा सकते हैं — वे अलग से मुहैया कराए जाएँगे और यह अध्याय इस गाइड के किसी भविष्य के संशोधन में पूरा किया जाएगा। इसे जानबूझकर सबसे आख़िर में, About के बाद रखा गया है, ठीक इसलिए कि बाद में इसे जोड़ने से किसी और अध्याय के पेज नंबर या सेक्शन संदर्भ न बदलें।

पहले से जो पता है और जो पूरे हो चुके अध्याय की नींव बनेगा: पैनल में एक साफ़ दिखने वाला **BETA** बैज है और शुरुआत में एक नोट है कि ड्राइवर ख़ुद मालिकाना है — Void द्वारा NVIDIA के आधिकारिक इंस्टॉलर से एक तय की गई चेकसम के साथ पैकेज किया गया, लेकिन स्वतंत्र रूप से जाँचा नहीं जा सकता, वही ईमानदारी जो Odyssey का बाक़ी हिस्सा ख़ुद पर लागू करता है। यह आपके ख़ास GPU को एक अंतर्निहित डेटाबेस से मिलाकर पहचानता है और सही ड्राइवर ब्रांच अपने आप चुनता है (current/Turing-और-नए, या पुराने कार्ड्स के लिए तीन लेगेसी ब्रांच में से एक, और जब कोई मालिकाना विकल्प लागू न हो तो ओपन-सोर्स ड्राइवर `nouveau` पर वापस लौटता है)। इंस्टॉलेशन एक दिखने वाले प्रगति संकेतक के साथ सात-चरणों की ट्रैक की गई पाइपलाइन के रूप में चलता है, और आपके इंस्टॉल पर क्लिक करने से पहले ही पैनल अपना ख़ुद का रिकवरी कमांड पहले से छाप देता है — एक टेक्स्ट कंसोल से `odyssey-nvidia --revert`। हाइब्रिड लैपटॉप पर, एक PRIME render-offload सेक्शन अपने आप दिखाई देता है, जो किसी ख़ास गेम या प्रोग्राम को NVIDIA GPU पर चलने देता है जबकि बाक़ी सब कुछ बैटरी की उम्र के लिए इंटीग्रेटेड GPU पर बना रहता है।