ODYSSEY LINUX · DOCUMENTAZIONE DI RIFERIMENTO

Guida alla sicurezza e all'hardening

Il firewall. AppArmor. I default di sistema.
E come verificare tutto quanto.

Launch Edition · Luglio 2026

Don't trust me. Verify me.
— nobody

Launch Edition. Si applica a Odyssey Linux con `linux-ether` 6.12 e l'Odyssey Control Center così come vengono distribuiti sulla ISO Launch Edition.

Copyright © 2026 il progetto Odyssey Linux.

Questo documento è rilasciato sotto licenza Creative Commons Attribuzione – Condividi allo stesso modo 4.0 Internazionale (CC BY-SA 4.0). Puoi copiarlo, ridistribuirlo e adattarlo, a condizione di attribuirne la fonte e di condividere le opere derivate sotto la stessa licenza.

Il sorgente di questo documento è pubblicato su `code.odysseylinux.org/nobody/odyssey-docs` e si compila in modo riproducibile con Typst. Il PDF che stai leggendo può essere verificato contro il repository — come ogni altra cosa in Odyssey.

Ogni file di configurazione, comando e impostazione predefinita citato in questa guida è stato letto dai sorgenti reali di Odyssey al momento della scrittura, non a memoria e non dalla documentazione di un'altra distribuzione.

NOTA SULLA LINGUA

Le etichette dell'interfaccia (`PROTECTED`, `SAVE & APPLY`, `ADD SUGGESTED RULE`…), i nomi dei pannelli e delle schede, i percorsi dei file, i comandi e l'output dei terminali restano in inglese: sono l'interfaccia reale del sistema, e tradurli renderebbe la guida inutilizzabile davanti allo schermo. Tutto il resto è tradotto.

INDICE

1 Il modello di sicurezza di Odyssey 4
1.1 Che cosa è attivo appena installato 4
1.2 Il modello di minaccia 5
1.3 Irrobustito ma modificabile 6
1.4 Come si legge questa guida 6
1.5 Gli strati a colpo d'occhio 7
2 Il firewall 8
2.1 Il firewall in una pagina 8
2.2 Perché nftables, e perché direttamente 8
2.3 Il ruleset predefinito, riga per riga 9
2.4 Verificarlo 10
2.5 Il pannello del Control Center 11
2.6 I preset: che cosa, quando e a quale rischio 13
2.7 Oltre la GUI: l'essenziale di nft 16
2.8 Ricette 17
2.9 Risoluzione dei problemi 17
3 AppArmor 19
3.1 Il controllo di accesso obbligatorio in parole semplici 19
3.2 Come lo distribuisce Odyssey 19
3.3 Anatomia di un profilo 20
3.4 Il pannello del Control Center 20
3.5 Il ciclo di confinamento 23
3.6 Oltre la GUI 24
3.7 Risoluzione dei problemi 25
4 I default di sistema 26
4.1 Il kernel: linux-ether 26
4.2 sysctl: come persiste la messa a punto 26
4.3 Privilegio sul desktop: polkit 26
4.4 Servizi in esecuzione per impostazione predefinita 27
4.5 Disciplina di aggiornamento 29
5 Verificami 30
5.1 Il problema della fiducia 30
5.2 La catena, da un capo all'altro 30
5.3 Verificare un pacchetto da solo 31
5.4 Un solo comando: odyssey-challenge 31
5.5 Il gate di scansione 31
5.6 Il cane da guardia 32
5.7 Le chiavi 33
6 Appendice 34
6.1 Riepilogo dei comandi 34
6.2 Glossario 34
6.3 Letture di approfondimento 35

CAPITOLO 1

Il modello di sicurezza di Odyssey

La maggior parte delle distribuzioni Linux si installa senza firewall, senza controllo di accesso obbligatorio e con il garbato suggerimento che prima o poi dovresti configurarteli da solo. Odyssey no. Il sistema che avvii dopo l'installazione è già difeso, e questo capitolo spiega esattamente che cosa è attivo, perché è stato scelto e dove stanno i confini di quella protezione.

Quest'ultima parte conta. Una documentazione di sicurezza che ti racconta solo da che cosa un sistema ti protegge è marketing. Un riferimento ti dice anche da che cosa **non** ti protegge, così puoi decidere che cosa aggiungere — e questa guida ti dà gli strumenti per aggiungerlo.

1.1 Che cosa è attivo appena installato

Dal primo avvio di un sistema Odyssey installato, senza che tu tocchi nulla:

Il firewall è acceso. nftables è installato, abilitato come servizio runit e caricato con un ruleset predefinito che scarta tutto il traffico in entrata non sollecitato. Nessuna porta è aperta. La tua macchina non risponde agli sconosciuti. Il capitolo 2 seziona il ruleset riga per riga.

AppArmor è armato. Il kernel si avvia con `apparmor=1 security=apparmor` sulla riga di comando, quindi il framework di controllo di accesso obbligatorio è caricato e osserva fin dall'inizio. I profili girano in modalità `complain` per impostazione predefinita: le violazioni vengono registrate, niente viene bloccato, e sei tu a decidere — dal Control Center — quando passare a `enforce`. Il capitolo 3 copre l'intero ciclo di vita.

L'elevazione dei privilegi è mediata. Le azioni amministrative dal desktop passano da polkit: una finestra grafica di autenticazione, una policy definita per ogni azione, nessuna shell di root permanente. Capitolo 4.

Ogni pacchetto è verificabile — e sorvegliato. I pacchetti nel repository Odyssey sono costruiti da ricette pubbliche, sottoposti a scansione antimalware prima della firma, firmati due volte — una firma RSA controllata da `xbps` a ogni installazione, più una firma cosign ancorata al registro pubblico di trasparenza Rekor — elencati in un manifesto firmato e riverificati ogni poche ore da un monitor esterno al server, che può avvisarti via email se qualcosa dovesse mai fallire. Non devi fidarti del manutentore; puoi controllarlo. Il capitolo 5 è il manuale per farlo davvero.

La cabina di regia di tutto questo è l'Odyssey Control Center — ogni protezione di questo elenco ha un suo pannello lì dentro, e ogni pannello legge e scrive gli stessi file di testo che questa guida ti insegna a modificare a mano.

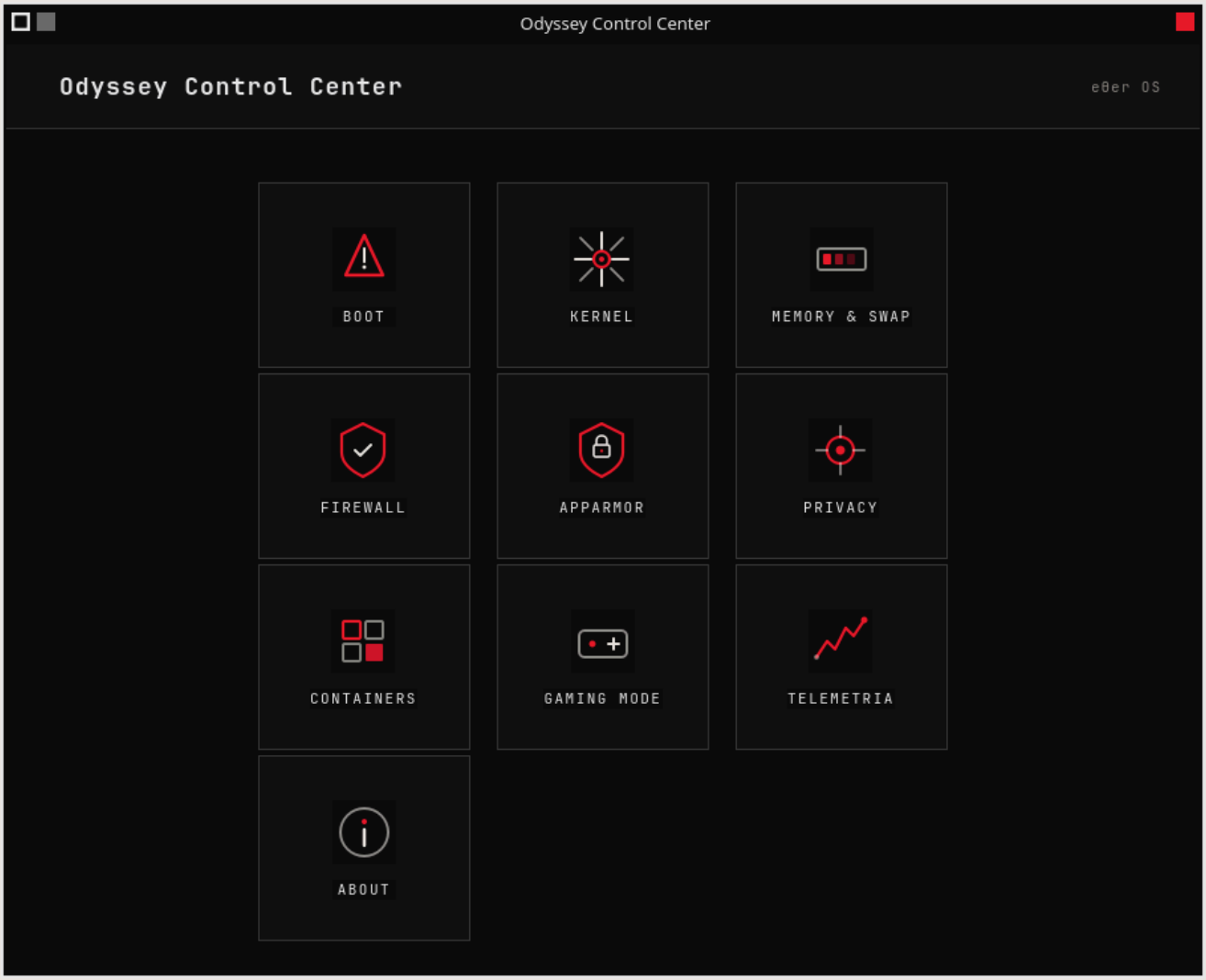

FIG · L'Odyssey Control Center. La seconda riga è il territorio di questa guida: Firewall, AppArmor, Privacy.

> **IL RIASSUNTO IN UNA RIGA**
> Chiuso verso la rete, sorvegliato dall'interno, amministrato attraverso un cancello e costruito allo scoperto. Tutto il resto di questa guida è dettaglio.

1.2 Il modello di minaccia

Una misura di sicurezza ha senso solo contro una minaccia dichiarata. I default di Odyssey sono tarati sulle minacce realistiche contro un sistema **desktop** — una macchina usata da una persona, spesso su reti che quella persona non controlla.

DA CHE COSA I DEFAULT TI DIFENDONO

Attacchi di rete opportunistici. La radiazione di fondo di internet: scanner automatici che sondano ogni indirizzo raggiungibile in cerca di porte aperte e servizi dimenticati. Su una rete di un bar o di un hotel, questo include la macchina accanto alla tua. Il firewall che scarta per impostazione predefinita rende Odyssey invisibile a tutta questa classe di attacchi — non c'è niente da sondare.

Applicazioni che si comportano male o compromesse. Un browser sfruttato da una pagina malevola, un lettore multimediale a cui viene dato in pasto un file costruito ad arte, un'app che semplicemente

esagera. AppArmor limita che cosa un programma profilato può leggere, scrivere ed eseguire, così la compromissione del programma non diventa la compromissione dei tuoi dati.

Manomissione della catena di distribuzione. Un pacchetto modificato, un mirror compromesso, una chiave del manutentore usata di nascosto. Doppia firma, checksum upstream fissati, ricette di build pubbliche e il registro di trasparenza Rekor rendono la manomissione rilevabile — da te, non solo da noi.

DA CHE COSA I DEFAULT NON TI DIFENDONO

Sezione di onestà. Nessuna configurazione predefinita protegge da queste cose:

Un attaccante fisico. Chi ha in mano il tuo portatile sbloccato è il tuo utente. Chi ha in mano il tuo portatile bloccato ma non cifrato può leggere il disco da una chiavetta live. La cifratura dell'intero disco in fase di installazione è nella coda di sviluppo di Odyssey; fino ad allora, considera il possesso fisico come partita persa.

Tu, che esegui codice non fidato con i tuoi privilegi. Se fai `curl | sh` di uno script preso da un forum, quello gira come te e può raggiungere tutto ciò che puoi raggiungere tu. Nessun firewall ferma il traffico che hai iniziato tu; il traffico in uscita è aperto per scelta progettuale (la sezione 2.3 spiega perché).

Un avversario mirato e ben finanziato. Se uno stato nazionale è specificamente interessato a te, i default di una distribuzione desktop non sono la tua soluzione — lo sono le tue pratiche operative. Odyssey ti dà un pavimento solido, non un programma di protezione testimoni.

1.3 Irrobustito ma modificabile

Alcuni progetti rispondono alle minacce qui sopra rendendo il sistema immutabile: un'immagine di base in sola lettura che non puoi modificare e che quindi non puoi rompere. Odyssey deliberatamente non lo fa.

Il ragionamento: un sistema immutabile ti protegge togliendoti le mani dai comandi. La posizione di Odyssey è che la macchina è tua. Ogni protezione descritta in questa guida può essere ispezionata, regolata o spenta — dal Control Center o da un editor di testo. I default sono opinionati; il sistema non è chiuso a chiave. Irrobustito, ma modificabile.

Questo mette un peso corrispondente sulla documentazione, ed è il motivo per cui questa guida esiste: se i comandi sono nelle tue mani, meriti di sapere con precisione che cosa fa ognuno di essi prima di toccarlo.

1.4 Come si legge questa guida

Ogni passo concreto di hardening in questo documento è presentato nello stesso blocco a quattro campi:

WHAT	L'azione esatta — un percorso di clic nel Control Center o un comando.
WHY	Il problema che affronta, in una o due frasi.
RESULT	Che cosa ci guadagni, formulato in modo che tu possa verificarlo da solo.
RISK	Che cosa si può rompere, chi ne è interessato e come si torna indietro.

Se sei esperto: leggi WHAT, scorri RISK, vai avanti. Se sei alle prime armi: leggi prima WHY e RISK — ti dicono se quel passo fa per te oppure no. Niente in questa guida è obbligatorio. Il testo di raccordo tra i blocchi porta la profondità: come funziona il meccanismo, quali erano le alternative e perché Odyssey distribuisce ciò che distribuisce.

I comandi mostrati con un prompt rosso `$` girano come tuo utente; quelli che richiedono i privilegi di root sono mostrati con `sudo`. Ogni file citato in questa guida è citato dal file reale così come viene distribuito

— se il tuo sistema installato è diverso, o l'hai cambiato tu, o la tua ISO è più recente di questo documento (controlla l'edizione in copertina).

1.5 Gli strati a colpo d'occhio

STRATO	MECCANISMO	CAP.
Rete	nftables, ruleset che scarta per default, supervisionato da runit	2
Applicazioni	Profili AppArmor — `complain` per default, `enforce` su richiesta	3
Privilegi	polkit / pkexec, policy per singola azione	4
Kernel	`linux-ether` 6.12 LTS, scheduler BORE, hardening in roadmap	4
Pacchetti	Gate di scansione, firme RSA + cosign, registro Rekor, manifesto firmato, monitor esterno al server	5

Ogni strato cede in modo indipendente. Un buco in uno non apre gli altri — che è poi il senso della stratificazione, e il motivo per cui questa guida tratta ciascuno nel proprio capitolo.

CAPITOLO 2

Il firewall

Odyssey viene distribuito con nftables installato, abilitato e caricato con un ruleset che scarta per impostazione predefinita, fin dal primo avvio. Questo capitolo spiega che cosa significa, percorre il ruleset predefinito una riga alla volta, documenta il pannello Firewall del Control Center e ognuno dei suoi preset, e poi va oltre la GUI fino alla riga di comando `nft` per i casi che il pannello non copre.

2.1 Il firewall in una pagina

Se sai già che cos'è un firewall stateful, salta a 2.2.

Ogni servizio di rete sulla tua macchina è in ascolto su una porta — una porta numerata. SSH ascolta sulla 22, un server web sulla 80 e sulla 443, e così via. Una connessione è **in entrata** quando qualcun altro bussa a una delle tue porte, e **in uscita** quando sei tu a bussare alle sue (caricare una pagina web, sincronizzare la posta).

Un firewall di host decide a quali bussate si risponde. Le due politiche oneste sono:

- **Accetta per default, blocca ciò che è cattivo.** Devi conoscere in anticipo ogni cosa cattiva. Non ci riuscirai.
- **Scarta per default, apri ciò che serve.** Tutto viene rifiutato a meno che tu non l'abbia aperto deliberatamente. È ciò che fa Odyssey.

Il perfezionamento che rende vivibile lo scarto predefinito è il **tracciamento delle connessioni** (filtraggio stateful): quando sei tu a iniziare una connessione, il firewall se la ricorda e lascia rientrare le risposte. Così una macchina chiusa per default naviga, si aggiorna e riproduce contenuti in streaming perfettamente — semplicemente non risponde mai a una conversazione che non ha iniziato.

2.2 Perché nftables, e perché direttamente

nftables è il framework nativo di filtraggio dei pacchetti del kernel Linux — il successore di iptables, nel kernel principale dalla versione 3.13. È ciò che filtra davvero i tuoi pacchetti su qualunque distribuzione moderna, indipendentemente dallo strumento dipinto sopra: `ufw` e `firewalld` sono livelli di gestione che generano regole per lo stesso motore sottostante.

Odyssey salta i livelli di gestione e parla direttamente al motore, per tre motivi:

1. **Un'unica fonte di verità.** Lo stato completo del firewall vive in un solo file leggibile, `/etc/nftables.conf`. Nessuno stato di demone, nessuna astrazione a zone, nessun livello di traduzione che possa dissentire dal file.
2. **Nessun demone in più.** `firewalld` vuole un servizio sempre in esecuzione e D-Bus. Su Odyssey il ruleset viene caricato una volta da un oneshot di runit e il kernel fa il resto.
3. **La GUI insegna il comando.** Il pannello del Control Center (2.5) legge e scrive lo stesso file che puoi aprire in un editor. Tutto ciò che fa il pannello, questo capitolo te lo mostra anche a mano. Non c'è nessuno strato nascosto tra il clic e il kernel.

Il lato servizio è puro runit: il servizio nftables è collegato nella directory canonica dei servizi predefiniti, così il ruleset viene caricato presto a ogni avvio:

```
/etc/runit/runsvdir/default/nftables → /etc/sv/nftables
```

2.3 Il ruleset predefinito, riga per riga

Questo è il contenuto completo di `/etc/nftables.conf` così come viene distribuito sulla ISO — non un estratto:

```
#!/usr/sbin/nft -f
# Odyssey Linux - default firewall ruleset
# Edit via Odyssey Control Center → Firewall, or directly here.

flush ruleset

table inet filter {
    chain input {
        type filter hook input priority filter; policy drop;

        # accept already-established connections (replies to our outbound)
        ct state established,related accept
        ct state invalid drop

        # loopback interface
        iif "lo" accept

        # ICMP (ping) - keeps diagnostics working, no real risk
        ip protocol icmp accept
        ip6 nexthdr icmpv6 accept
    }

    chain forward {
        type filter hook forward priority filter; policy drop;
    }

    chain output {
        type filter hook output priority filter; policy accept;
    }
}
```

Leggendolo dall'alto in basso:

`flush ruleset`	si riparte da zero. Caricare questo file sostituisce qualunque regola fosse attiva, in modo atomico — il kernel scambia l'intero ruleset in una sola transazione, così non esiste mai un istante con mezzo firewall.
`table inet filter`	una sola tabella per IPv4 e IPv6. La famiglia `inet` significa che ogni regola qui sotto si applica a entrambi i protocolli contemporaneamente — una macchina protetta solo su IPv4 è mezzo firewall.
`chain input … policy drop`	il cuore del progetto. Questa catena vede ogni pacchetto indirizzato alla tua macchina, e il suo verdetto predefinito è `drop` — scarta in silenzio. Non «rifiuta con un errore»: scarta. Uno scanner che ti sonda non riceve alcuna risposta, il che è indistinguibile da un indirizzo vuoto.
`ct state established,related accept`	la regola di tracciamento delle connessioni di 2.1. Le risposte alle connessioni che hai iniziato tu (`established`) e il traffico di supporto a esse legato (`related`, per esempio i canali dati di FTP) rientrano. Questa singola regola è il motivo per

	cui una macchina completamente chiusa ha comunque internet perfettamente funzionante.
`ct state invalid drop`	pacchetti che non appartengono ad alcuna connessione nota e che sono malformati come inizio di conversazione — frammenti fuori sequenza, artefatti di scansione. Scartati esplicitamente.
`iif «lo» accept`	l'interfaccia di loopback. Programmi sulla tua macchina che parlano con altri programmi sulla tua macchina (`127.0.0.1`, `::1`) — database, server di sviluppo, audio, metà di un desktop moderno. Mai filtrata.
`ICMP accettato, v4 e v6`	la tua macchina risponde al ping e, cosa cruciale, elabora i messaggi di path-MTU e di errore. Su IPv6 ICMPv6 non è opzionale — la neighbour discovery vive lì, e filtrarla rompe il protocollo. Rispondere al ping rivela solo che un indirizzo è in uso, e il valore diagnostico supera di gran lunga questo dato.
`chain forward … policy drop`	traffico che attraversa la tua macchina da una rete a un'altra. Un desktop non è un router; niente viene inoltrato. (Il software di container e di virtualizzazione che hanno bisogno dell'inoltro inseriscono le proprie regole a runtime.)
`chain output … policy accept`	il tuo traffico in uscita non è filtrato. È un compromesso desktop deliberato: il filtraggio in uscita aiuta solo contro malware già in esecuzione sulla tua macchina, al prezzo di rompere il software in modo imprevedibile per tutti gli altri. Se sulla tua macchina sta girando codice che non hai autorizzato, il firewall è lo strato sbagliato per combatterlo — quella battaglia è del capitolo 3.

> **LA CONSEGUENZA PRATICA**
> Appena installato: nessuna porta risponde, niente di ciò che esegui localmente viene disturbato, tutto ciò che inizi tu funziona, il ping funziona. Devi toccare il firewall soltanto quando questa macchina deve offrire un servizio ad altre macchine — ed è di questo che parla il resto del capitolo.

2.4 Verificarlo

Non fidarti di niente, nemmeno di questo capitolo. Controlla il firewall sul tuo sistema:

WHAT	`sudo nft list ruleset`
WHY	Leggere `/etc/nftables.conf` mostra le intenzioni; questo mostra le regole realmente caricate nel kernel in questo momento.
RESULT	Output corrispondente alla sezione 2.3. Se è vuoto, il firewall non è attivo — vedi la risoluzione dei problemi, 2.9.
RISK	Nessuno. Sola lettura.

E controlla che cosa sarebbe raggiungibile se il firewall non ci fosse:

WHAT	`ss -tlnp` (in ascolto su TCP) e `ss -ulnp` (UDP)
WHY	Il firewall protegge i servizi in ascolto. Sapere che cosa è in ascolto — e su quali indirizzi — ti dice che cosa la politica di scarto sta effettivamente proteggendo.

RESULT	Un elenco di socket in ascolto. Le voci legate a `127.0.0.1` o `::1` sono solo locali e irraggiungibili dall'esterno indipendentemente dal firewall; le voci su `0.0.0.0` o `*` sono ciò che la politica di scarto sta proteggendo.
RISK	Nessuno. Sola lettura.

2.5 Il pannello del Control Center

Apri Odyssey Control Center → Firewall. Il pannello è un front-end per `/etc/nftables.conf`: analizza il file all'apertura, tiene in memoria le tue modifiche come bozza e scrive soltanto dietro tua conferma esplicita.

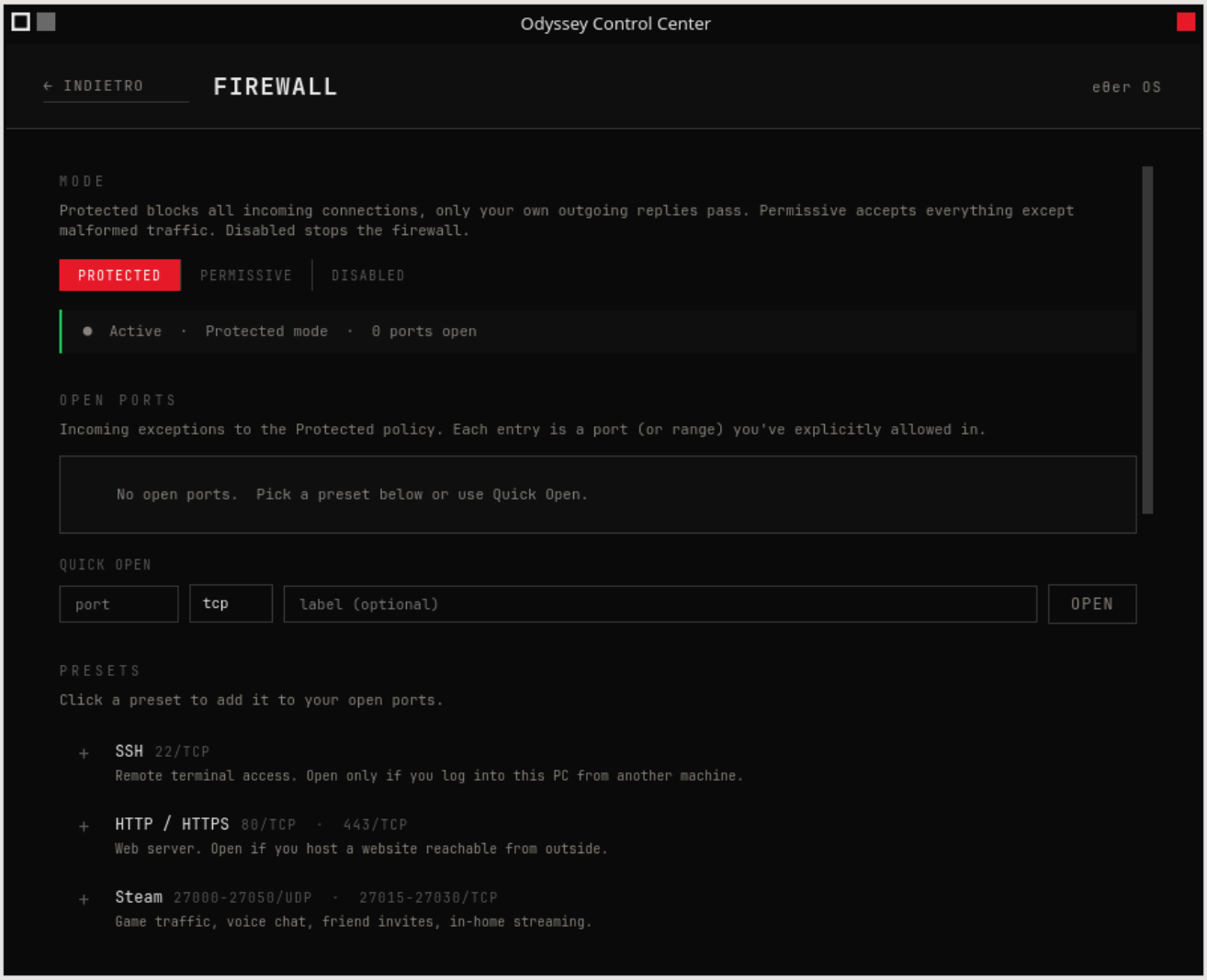

FIG · Il pannello firewall nel suo stato predefinito: PROTECTED, zero porte aperte, la riga di stato che conferma il ruleset caricato.

Dall'alto in basso:

MODE — tre posizioni. `PROTECTED` è il default descritto in 2.3. `PERMISSIVE` tiene il servizio in esecuzione ma accetta il traffico in entrata — è l'impostazione diagnostica per «è il firewall che sta rompendo questa cosa?», non un posto dove vivere. `DISABLED` scarica il ruleset e ferma il servizio.

OPEN PORTS — le eccezioni che hai concesso, ognuna con il suo nome, le sue porte e una × per revocarla. Vuoto per default, e sulla maggior parte dei desktop dovrebbe restare così.

QUICK OPEN — porta, protocollo, etichetta. Per l'utente esperto il cui servizio non è tra i preset; l'etichetta mantiene la voce autoesplicativa anche mesi dopo.

PRESETS — 8 servizi comuni a un clic di distanza, altri 13 sotto `MORE PRESETS`. Ogni preset porta con sé una descrizione in una riga di quando lo vorresti — il pannello è progettato per essere leggibile **prima** che cliccabile. La sezione 2.6 estende ognuna di quelle descrizioni con il lato del rischio.

ADVANCED → VIEW RAW CONFIG — il file vero e proprio, in un editor. Se la tua configurazione è stata scritta a mano in una forma che il parser non riconosce, il pannello si fa da parte e ti offre **soltanto** l'editor grezzo invece di tirare a indovinare.

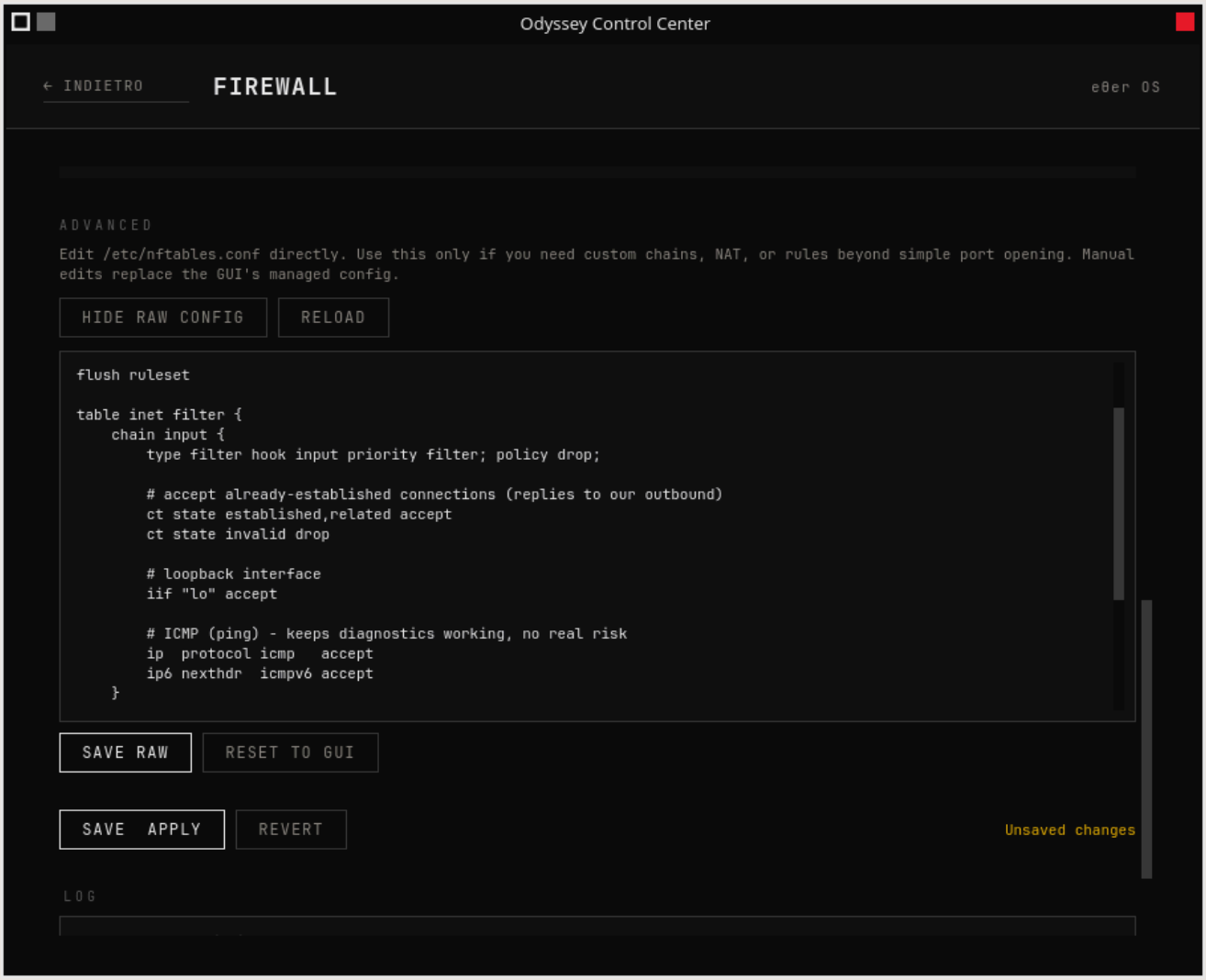

FIG · VIEW RAW CONFIG: lo stesso ruleset predefinito della sezione 2.3, modificabile sul posto. SAVE RAW scrive il tuo testo; RESET TO GUI riporta alla configurazione gestita.

IL FLUSSO CONFERMA/SALVATAGGIO

Niente di ciò che clicchi ha effetto immediato. Le modifiche si accumulano come un modello in bozza — il pannello mostra un indicatore giallo **Unsaved changes** — finché non premi `SAVE & APPLY`, che in una sola operazione privilegiata: fa il backup del file corrente in `/etc/nftables.conf.bak`, scrive il nuovo ruleset, lo carica atomicamente con `nft -f` e allinea il servizio runit alla modalità che hai scelto. `REVERT` scarta le modifiche in bozza.

Non è burocrazia. Un firewall applicato regola per regola può tagliare la connessione attraverso cui lo stai amministrando — il modo classico di chiudersi fuori da una macchina remota via SSH. Metti in bozza, rivedi, applica una volta sola.

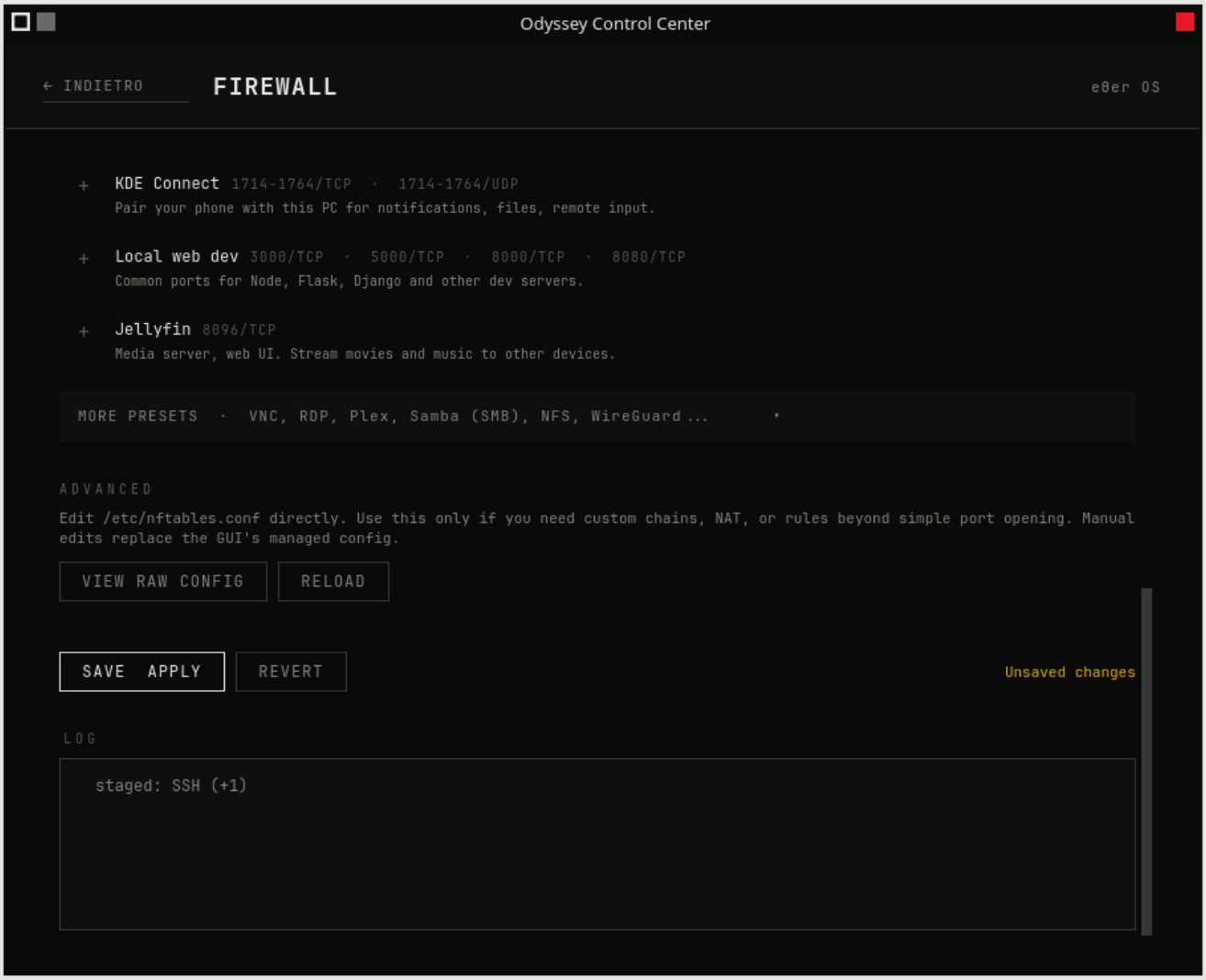

FIG · Un preset in bozza ("staged: SSH (+1)" nel log), l'indicatore giallo Unsaved changes e SAVE & APPLY / REVERT. Il kernel non è ancora stato toccato.

2.6 I preset: che cosa, quando e a quale rischio

Aprire una porta è una dichiarazione: **qui eseguo un servizio, e macchine che non controllo possono parlarci.** Da quel momento, la sicurezza della tua macchina su quella porta è pari alla sicurezza di quel servizio — la sua autenticazione, il suo livello di aggiornamento, la sua configurazione. Il compito del firewall era tenerti fuori da quella faccenda; ogni preset qui sotto è la decisione documentata di entrarci.

Due principi valgono per tutti:

1. **Non aprire nulla che non stai attivamente servendo.** Una porta aperta senza nessuno in ascolto è innocua ma inutile; una porta aperta con un servizio dimenticato e non aggiornato è il modo in cui le macchine cadono.
2. **Il tuo firewall non è il tuo unico muro.** Su una rete domestica dietro NAT, una porta aperta è tipicamente raggiungibile solo dalla tua LAN, a meno che tu non la inoltri anche sul router. Su una rete pubblica o universitaria, dai per scontato che ogni porta aperta sia raggiungibile da estranei.

PRESET RAPIDI

SSH `22/tcp`
Accesso remoto al terminale. Aprilo solo se accedi a questa macchina da un'altra. La porta più utile — e più attaccata — di qualunque sistema Linux.

RISK SSH esposto su internet viene sottoposto a forza bruta nel giro di minuti, in permanenza, dai bot. Prima di aprirlo: disabilita l'accesso con password in favore delle chiavi (`PasswordAuthentication no` in `/etc/ssh/sshd_config`) e valuta di restringere la regola alla tua LAN (2.8). Con le sole chiavi SSH è temprato dalla battaglia; con le password è un gioco di indovinelli che ospiti tu.

HTTP / HTTPS `80, 443/tcp`

Ospiti un sito o un'applicazione web che deve essere raggiungibile da altre macchine.

RISK Ora stai eseguendo un server web pubblico: le sue CVE sono le tue CVE, e tutto ciò che serve è a una configurazione sbagliata dall'essere pubblico. Tieni il server aggiornato, servi TLS sulla 443 e non puntarlo mai a una directory che contenga qualcosa di privato.

Steam `27000-27050/udp, 27015-27030/tcp`

Traffico di gioco, chat vocale, inviti agli amici e streaming in casa funzionano in modo più affidabile con questi intervalli aperti — in particolare ospitare lobby e trasmettere ad altri dispositivi.

RISK Basso. Gli intervalli sono ampi ma contano solo mentre Steam è in ascolto su di essi. Chiudi il preset se smetti di usare Steam; un ampio intervallo dormiente è esattamente il tipo di cosa che ci si dimentica.

Minecraft (Java) `25565/tcp`

Ospiti un server Minecraft Java Edition a cui gli amici si connettono da altre macchine.

RISK Un server Minecraft pubblico è un server vero: verrà trovato dagli scanner nel giro di ore e sondato alla ricerca di exploit noti. Tieni aggiornato il software del server, abilita la whitelist e chiudi la porta quando la mappa si raffredda.

Syncthing `22000/tcp+udp, 21027/udp`

Sincronizzazione di file peer-to-peer, cifrata end-to-end, tra i tuoi dispositivi. La 21027 è la scoperta sulla rete locale.

RISK Modesto per costruzione — Syncthing autentica i dispositivi reciprocamente e cifra tutto. Il rischio residuo è il software in sé; ha un buon curriculum, ma è comunque un servizio in ascolto e va tenuto aggiornato come tale.

KDE Connect `1714-1764/tcp+udp`

Accoppia il telefono con questa macchina: notifiche, trasferimento file, appunti, input remoto.

RISK Cinquanta porte sono un'apertura grande, e il modello di accoppiamento di KDE Connect si fida della rete locale. Va bene a casa; sul Wi-Fi pubblico significa che qualunque dispositivo intorno a te può tentare l'accoppiamento. Restringi alla LAN (2.8) oppure spegni il preset quando sei lontano da reti fidate.

Local web dev `3000, 5000, 8000, 8080/tcp`

Le porte classiche di Node, Flask, Django e compagnia — aprile quando devi raggiungere un server di sviluppo da un altro dispositivo, tipicamente un telefono che prova un layout.

RISK I server di sviluppo esplicitamente non sono irrobustiti contro il traffico ostile — endpoint di debug, errori verbosi, a volte esecuzione di codice per progetto. Apri solo su reti fidate, e tratta questo preset come qualcosa che accendi per un pomeriggio, non come uno stato permanente.

Jellyfin `8096/tcp`

La tua macchina è un server multimediale Jellyfin che trasmette a TV, telefoni e altri dispositivi; la 8096 è l'interfaccia web e l'API.

RISK L'interfaccia web è una porta protetta da login verso tutta la tua libreria e verso un software server con una storia di CVE ben viva. Sulla LAN è sensato; attraverso internet, mettilo dietro un reverse proxy con TLS o una VPN invece di esporre la 8096 nuda.

ALTRI PRESET

VNC `5900/tcp`

Desktop remoto via VNC — vedi e controlli lo schermo di questa macchina da altrove.

RISK Il VNC classico non ha cifratura del trasporto: password e contenuto dello schermo attraversano la rete in chiaro. Non esporre la 5900 oltre una LAN fidata; per qualunque altra cosa, incapsulala in SSH o WireGuard. L'accesso allo schermo è accesso totale.

RDP `3389/tcp`

Desktop remoto in stile Windows (`xrdp` su Linux).

RISK La 3389 è una delle porte più scansionate di internet, con una lunga storia di exploit critici. Solo LAN o VPN — una porta RDP esposta su internet è un invito permanente.

Plex `32400/tcp`

Server multimediale Plex, interfaccia web ed endpoint di streaming.

RISK Stessa forma di Jellyfin: una porta con login verso la tua libreria, su software server attivamente mantenuto. Preferisci la LAN o l'accesso remoto tramite relay di Plex all'esposizione diretta; tienilo aggiornato.

Samba (SMB) `139, 445/tcp · 137, 138/udp`

Condivisione file compatibile con Windows sulla rete locale.

RISK SMB ha una delle storie di exploit più ricche dell'informatica — la 445 è la porta su cui si è diffuso WannaCry. Non aprirla mai verso internet in nessuna circostanza; restringi alla LAN, tieni le condivisioni protette da password e preferisci le versioni moderne del protocollo (SMB3) nella tua configurazione di Samba.

NFS `2049/tcp`

Network File System — condivisioni da Linux a Linux, tipicamente verso un NAS o tra le tue macchine.

RISK L'NFS classico si fida della rete: le esportazioni sono comunemente filtrate per indirizzo IP e il traffico non è cifrato. È rigorosamente un protocollo da LAN fidata; limita `/etc/exports` a indirizzi specifici, non a intere sottoreti che condividi con estranei.

WireGuard `51820/udp`

Questa macchina è un endpoint VPN a cui altri dispositivi si connettono — per esempio per raggiungere la rete di casa dal portatile all'estero.

RISK Il tipo buono di porta aperta. WireGuard non risponde affatto ai pacchetti non autenticati, quindi per uno scanner la porta sembra chiusa. Il rischio vero sta dietro: un peer la cui chiave è compromessa è dentro la tua rete. Custodisci le chiavi private.

PostgreSQL `5432/tcp`

Un'altra macchina deve raggiungere un database PostgreSQL in esecuzione qui — un application server, un collega sulla LAN.

RISK La porta di un database è una linea diretta verso i tuoi dati; i database esposti con credenziali deboli vengono raccolti su scala industriale. Apri solo verso reti fidate, richiedi le password (`scram-sha-256`) e limita `pg_hba.conf` a indirizzi di origine specifici. Se solo applicazioni locali usano il database, è già raggiungibile sul loopback — questo preset non ti serve affatto.

MySQL / MariaDB `3306/tcp`

Stesso scenario di PostgreSQL, per la famiglia MySQL.

RISK Calcolo identico: mai esposto su internet, associa e concedi privilegi per host specifico, credenziali robuste. Il loopback non richiede nessuna porta aperta.

Mosh `60000-61000/udp`

Mobile shell — un compagno di SSH che sopravvive al roaming e ai collegamenti instabili. Richiede SSH (si autentica attraverso di esso) più questo intervallo UDP.

RISK L'intervallo è ampio ma la porta di ogni sessione è protetta da una chiave crittografica di sessione scambiata via SSH. L'esposizione effettiva è vicina a quella di SSH con le chiavi; l'unica avvertenza è che una regola da mille porte sembra allarmante in un audit finché non è etichettata.

Source games `27015/tcp+udp`

Ospitare Counter-Strike, Team Fortress o altri server basati su motore Source.

RISK I server di gioco sono esposti a internet per definizione e vengono sondati come qualunque server. Tienili aggiornati e chiudi la porta quando non ospiti.

Minecraft Bedrock `19132/udp`

Ospitare un server Minecraft Bedrock Edition.

RISK Come il preset Java: server vero, scansioni vere, tienilo aggiornato, metti i giocatori in whitelist, chiudi quando è inattivo.

MPD `6600/tcp`
Music Player Daemon controllato da client su altri dispositivi.
RISK MPD non ha un'autenticazione degna di questo nome — chiunque possa raggiungere la 6600 controlla la riproduzione e può sfogliare i percorsi della tua musica. Rigorosamente LAN, e idealmente associato all'indirizzo di LAN anche in `mpd.conf`.

MQTT (Mosquitto) `1883/tcp`
Un broker MQTT per la messaggistica IoT — sensori e dispositivi di domotica che pubblicano verso questa macchina.
RISK La porta 1883 è MQTT in chiaro. Un broker aperto e non autenticato lascia trapelare ogni messaggio e ne accetta di iniettati — e migliaia di broker simili sono indicizzati pubblicamente. Richiedi come minimo nome utente e password in Mosquitto; per qualunque cosa oltre la LAN, usa TLS sulla 8883.

2.7 Oltre la GUI: l'essenziale di nft

Tutto quanto sopra, a mano. La grammatica è piccola: le tabelle contengono catene, le catene contengono regole, e una regola è una corrispondenza seguita da un verdetto.

Ispeziona il ruleset attivo (con gli handle delle regole, necessari per la cancellazione):

```
$ sudo nft -a list ruleset
```

Aggiungi una regola a runtime — questo apre la TCP 22 nel kernel in esecuzione, senza toccare il file:

```
$ sudo nft add rule inet filter input tcp dport 22 accept
```

Cancellala di nuovo tramite il suo handle:

```
$ sudo nft -a list chain inet filter input
  ... tcp dport 22 accept # handle 42
$ sudo nft delete rule inet filter input handle 42
```

> **RUNTIME CONTRO PERSISTENTE — LA DISTINZIONE CHE CONTA**
> `nft add` cambia il kernel adesso e sopravvive fino al riavvio o al successivo `nft -f`. `/etc/nftables.conf` è ciò che viene caricato all'avvio. Il Control Center tiene sempre i due allineati; quando lavori a mano, il meccanismo di allineamento sei tu. Una regola aggiunta dal vivo ma mai scritta nel file sparisce dopo il riavvio — il che è una funzionalità per gli esperimenti e un bug per i servizi.

Per rendere persistente una modifica a mano: modifica `/etc/nftables.conf`, poi carica l'intero file in modo atomico:

WHAT	`sudo nft -f /etc/nftables.conf` — dopo `sudo cp /etc/nftables.conf /etc/nftables.conf.bak`
WHY	Applica il file modificato in un'unica transazione del kernel: o l'intero nuovo ruleset è attivo, oppure (in caso di qualsiasi errore di sintassi) non è cambiato assolutamente nulla.
RESULT	Stato del firewall persistente e identico al file; verifica con `sudo nft list ruleset`.
RISK	Un errore di logica (a differenza di un errore di sintassi) si applica senza problemi e può comunque tagliarti fuori — su una macchina remota, tieni sempre aperta la sessione SSH corrente e prova una nuova connessione prima di chiuderla. Ripristino: `sudo nft -f /etc/nftables.conf.bak`.

2.8 Ricette

APRIRE UN SERVIZIO SOLO ALLA TUA LAN

I preset aprono le porte verso qualunque origine. Per i servizi che non hanno alcuna ragione di esistere oltre la tua rete — Samba, MPD, KDE Connect su un portatile — limita la regola all'intervallo di indirizzi della tua LAN modificando la regola nella configurazione grezza:

```
# invece di: tcp dport 445 accept
ip saddr 192.168.1.0/24 tcp dport 445 accept
```

Sostituisci `192.168.1.0/24` con la tua sottorete reale (`ip -4 addr` te la mostra). Il servizio ora non esiste per nessuno fuori da quell'intervallo — inclusi tutti quelli sul Wi-Fi del bar, dove la «LAN» del tuo portatile è una stanza piena di estranei su una sottorete diversa.

APRIRE UNA PORTA PER UN POMERIGGIO

Per il caso del server di sviluppo visto dal telefono, salta del tutto la persistenza:

```
$ sudo nft add rule inet filter input tcp dport 8000 accept
  # ... prova dal telefono ...
$ sudo nft -f /etc/nftables.conf   # ripristina lo stato del file = chiude
```

VEDERE LA TUA MACCHINA COME LA VEDE LA RETE

Da un altro dispositivo sulla stessa rete:

```
$ nmap IP_DELLA_TUA_ODYSSEY
```

Su un'installazione Odyssey predefinita, il risultato onesto è il silenzio — ogni porta filtrata. Qualunque cosa `nmap` riporti come aperta è la tua esposizione reale e attuale; dovrebbe corrispondere esattamente al tuo elenco `OPEN PORTS`. Se non corrisponde, scopri perché prima di fare qualunque altra cosa.

2.9 Risoluzione dei problemi

Un servizio non è raggiungibile da un'altra macchina. Procedi verso l'esterno: è effettivamente in ascolto (`ss -tlnp` — e su quale indirizzo? `127.0.0.1` significa solo locale, nessuna regola di firewall può aiutare)? La porta è aperta (`sudo nft list ruleset`)? Funziona dalla macchina stessa (`curl localhost:PORTA`)? Solo a quel punto dai la colpa alla rete che c'è in mezzo.

Chiuso fuori da SSH su una macchina remota. La prevenzione batte la cura: il flusso «metti in bozza e applica» del Control Center esiste per questo, e l'equivalente manuale è la regola delle due sessioni — applica la modifica in una sessione SSH, prova con una seconda sessione nuova, e solo allora chiudi la prima. Se è già troppo tardi, ti serve un accesso fuori banda (console fisica, o la console web del tuo provider VPS) per eseguire `sudo nft -f /etc/nftables.conf.bak`.

Il firewall risulta inattivo dopo l'avvio. Controlla il servizio e il suo collegamento:

```
$ sudo sv status nftables
$ ls -l /etc/runit/runsvdir/default/nftables
```

Il collegamento simbolico deve puntare a `/etc/sv/nftables`. Se invece le regole sono caricate ma inattese, ricorda che i motori di container e i client VPN legittimamente inseriscono le proprie tabelle a runtime — `nft list ruleset` mostra tutto, non solo la tabella `inet filter` di Odyssey.

Il pannello mostra solo l'editor grezzo. Non è un bug: il tuo `/etc/nftables.conf` non corrisponde più alla struttura che il pannello gestisce (tabelle scritte a mano, catene personalizzate). Il pannello si rifiuta di reinterpretare una configurazione che non ha plasmato lui, e ti consegna l'editor. Le tue regole non sono state toccate.

CAPITOLO 3

AppArmor

Il firewall decide chi può parlare con la tua macchina. AppArmor decide che cosa possono fare i programmi sulla tua macchina una volta che sono in esecuzione. Questo capitolo spiega il controllo di accesso obbligatorio in parole semplici, documenta come Odyssey lo distribuisce — armato all'avvio, in osservazione per impostazione predefinita — e poi documenta ciò che nessun'altra distribuzione ti dà appena installata: un flusso di confinamento completamente grafico, dallo scaricamento di un profilo alla trasformazione di una violazione registrata in una regola di profilo con un clic.

3.1 Il controllo di accesso obbligatorio in parole semplici

I permessi Unix classici rispondono a una sola domanda: **che cosa può fare questo utente?** Ogni programma che lanci eredita l'intera risposta. Il tuo browser, eseguito da te, può leggere le tue chiavi SSH, i tuoi documenti fiscali e la tua libreria di foto — non perché ne abbia bisogno, ma perché puoi tu.

È questa la falla sfruttata praticamente da ogni storia del tipo «un'app è stata compromessa»: l'attaccante non ottiene «il browser», ottiene **te**.

Il controllo di accesso obbligatorio (MAC) aggiunge una seconda domanda, più stretta: **che cosa può fare questo programma, chiunque lo esegua?** AppArmor risponde con profili per singolo programma — elenchi di autorizzazione dei file che un programma può leggere, scrivere ed eseguire, e delle capability che può usare. Un browser profilato che venga sfruttato può ancora essere il tuo browser; non può più essere le tue chiavi SSH.

AppArmor è uno dei due sistemi MAC mainstream di Linux. L'altro, SELinux, etichetta ogni oggetto del filesystem ed è notoriamente spietato da amministrare. AppArmor confina per percorso, e i suoi profili sono file che un essere umano può leggere dall'inizio alla fine — il che si sposa con una distribuzione la cui regola fondativa è che tu debba poter verificare tutto da solo.

3.2 Come lo distribuisce Odyssey

Tre fatti definiscono lo stato predefinito di un'installazione Launch Edition:

Il kernel è armato. La ISO scrive `apparmor=1 security=apparmor` nella riga di comando del kernel in GRUB, così ogni sistema Odyssey si avvia con l'LSM AppArmor attivo. È la parte che la maggior parte delle distribuzioni lascia a te, ed è la parte che richiede un riavvio per cambiare — motivo per cui Odyssey la fa prima del primo avvio.

Lo spazio utente è a bordo. Il pacchetto `apparmor` — parser, strumenti e l'insieme di profili standard — è presente sulla ISO, insieme al servizio runit che carica i profili all'avvio. Non c'è nulla da installare prima che il pannello del Control Center sia pienamente operativo.

I profili osservano, non bloccano. La modalità all'avvio è impostata in un unico file, `/etc/default/apparmor`, e viene distribuita come `APPARMOR=complain`: ogni azione che un profilo vieterebbe viene registrata e permessa. Niente si rompe, e il registro accumula silenziosamente la prova di che cosa avrebbe fatto l'applicazione delle regole.

È un progetto deliberato a due stadi. Il primo giorno non perdi nulla e guadagni una traccia di controllo. Quando **tu** decidi che un profilo si è dimostrato valido — il suo registro è silenzioso, oppure è pieno di cose che sei contento sarebbero state bloccate — lo porti a `enforce`, un profilo alla volta, dal pannello. Irrobustito ma modificabile, applicato al MAC.

> **LA MODALITÀ COMPLAIN NON È UN PLACEBO**
> Anche senza bloccare nulla, la modalità `complain` è la tua ricognizione: il registro delle violazioni ti dice esattamente che cosa tocca ogni programma profilato oltre il proprio profilo. Leggerlo è il modo in cui impari a conoscere il tuo sistema — e, in Odyssey, è letteralmente a un clic dal diventare la soluzione (3.5).

3.3 Anatomia di un profilo

I profili vivono in `/etc/apparmor.d/`, un file per ogni programma confinato. Uno minimale si legge così:

```
#include <tunables/global>

/usr/bin/example {
  #include <abstractions/base>

  /usr/bin/example        mr,     # legge + mappa se stesso
  /etc/example.conf       r,      # legge la sua configurazione
  owner @{HOME}/.example/ rw,     # la sua dotfile dir, solo proprietario
  /var/log/example.log    w,      # scrive il suo log

  network inet stream,            # TCP su IPv4
  capability net_bind_service,    # associa una porta privilegiata
}
```

La grammatica in un fiato: ogni riga è un percorso e i permessi concessi su di esso — `r` lettura, `w` scrittura, `x` esecuzione (in diverse varianti), `m` mappatura in memoria, `k` blocco. `owner` restringe una regola ai file di proprietà dell'utente del processo. Le righe `network` e `capability` concedono tipi di socket e capability del kernel per nome. Le **abstractions** sono file di inclusione condivisi per esigenze comuni (runtime di base, DNS, audio), così i profili non si ripetono. Tutto ciò che non è elencato è negato — un profilo è un elenco di autorizzazione, mai un elenco di blocco.

3.4 Il pannello del Control Center

Odyssey Control Center → AppArmor. Cinque schede — `STATUS`, `PROFILES`, `COMMUNITY`, `VIOLATIONS`, `EDITOR` — che insieme coprono l'intero ciclo di vita di un profilo senza uscire dalla GUI.

STATUS — LA VERITÀ SUL CAMPO

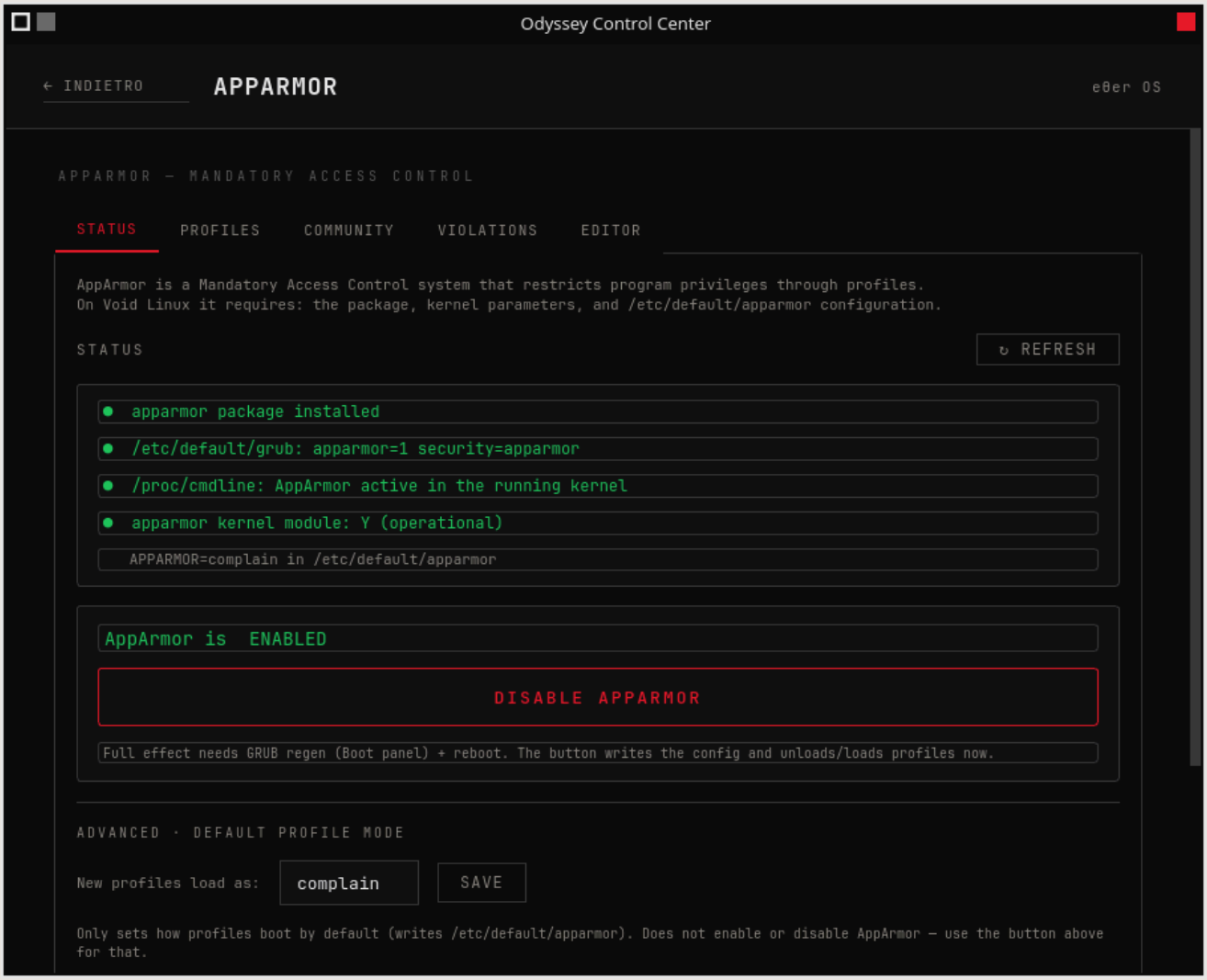

FIG · La scheda STATUS su un'installazione predefinita: tutti e quattro i controlli verdi — pacchetto, parametri GRUB, kernel in esecuzione, modulo del kernel — e la modalità predefinita dei profili letta da `/etc/default/apparmor`.

`STATUS` verifica l'intera pila, nell'ordine: il pacchetto `apparmor` (rilevato onestamente, dalla presenza dello strumento `aa-status`); i parametri `apparmor=1 security=apparmor` in `/etc/default/grub`; se il kernel **in esecuzione** si è davvero avviato con quei parametri (letto da `/proc/cmdline` — è questo il controllo che resta ambra, «riavvio necessario», quando i parametri sono scritti ma l'avvio corrente li precede); e infine il modulo del kernel stesso. L'ultima riga mostra la modalità predefinita all'avvio, presa da `/etc/default/apparmor`.

Qui vivono due controlli. Il grande pulsante abilita o disabilita il framework nel suo complesso — e **disabilitare significa disabilitare**: rimuove i parametri del kernel invece di lasciare un sistema mezzo armato che dichiara di essere spento. Il selettore di modalità in basso cambia soltanto `/etc/default/apparmor` — cioè come i profili vengono caricati all'avvio — senza toccare nient'altro.

> **UN PASSO MANUALE DELIBERATO**
> Le modifiche ai parametri del kernel vengono scritte in `/etc/default/grub`, ma il pannello non esegue mai `grub-mkconfig` alle tue spalle — rigenerare la configurazione del bootloader è una **tua** decisione, dal pannello Boot. Uno strumento che riscrive silenziosamente il tuo menu di avvio è uno strumento di cui

non puoi fidarti su una macchina in dual-boot; il pannello ti dice quando il passo serve e lascia il grilletto nella tua mano.

PROFILES — CONTROLLO PER SINGOLO PROFILO

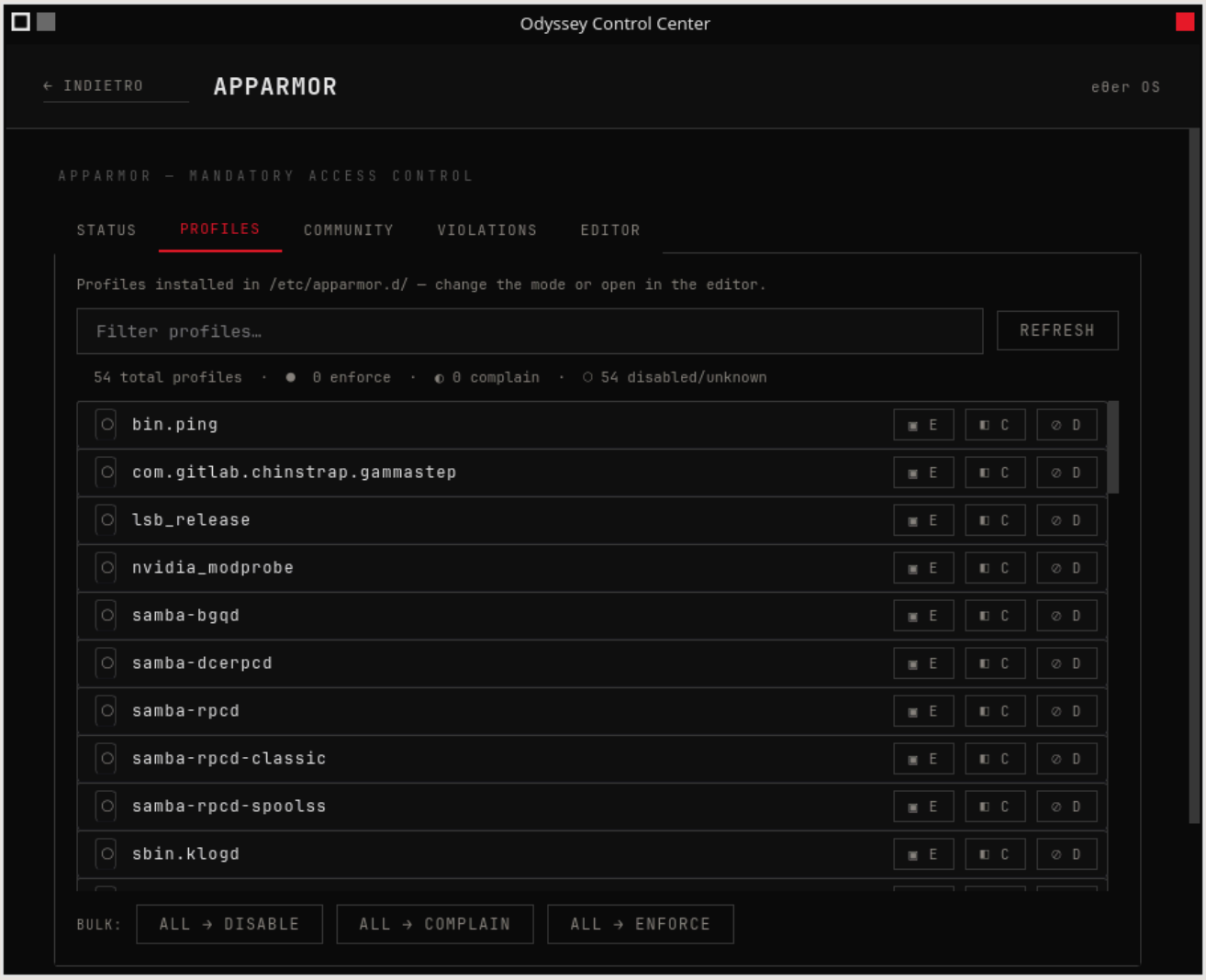

FIG · La scheda PROFILES: ogni profilo in /etc/apparmor.d/, un filtro, gli interruttori E/C/D per profilo e le azioni di massa in fondo.

Ogni profilo caricato in una riga, con la sua modalità corrente e tre interruttori — Enforce, Complain, Disable — più i pulsanti di massa per spostare l'intero insieme in una volta. Dietro l'interruttore, il pannello fa l'operazione come si deve: dopo `aa-enforce`, il profilo viene anche ricaricato nel kernel con `apparmor_parser -r`, così il cambiamento è immediatamente attivo — il classico tranello da riga di comando di un `enforce` che silenziosamente non cambia nulla qui non esiste.

COMMUNITY — PROFILI CHE NON HAI DOVUTO SCRIVERE

AppArmor standard confina soprattutto componenti di sistema. I programmi che ti preoccupano davvero — il browser, il client torrent, l'app di chat: tutto ciò che mangia input non fidato per mestiere — hanno bisogno di profili dalla comunità, e questa scheda li scarica direttamente dal repository ufficiale upstream dei profili AppArmor. Un insieme consigliato per l'uso desktop è preselezionato — browser, posta, visualizzatori di documenti, lettori multimediali, client torrent, messaggistica — e tutto ciò che installi atterra, come ogni profilo su Odyssey, in modalità `complain`: prima osserva, e si guadagna l'applicazione delle regole attraverso il ciclo qui sotto.

VIOLATIONS ED EDITOR — CHI CHIUDE IL CERCHIO

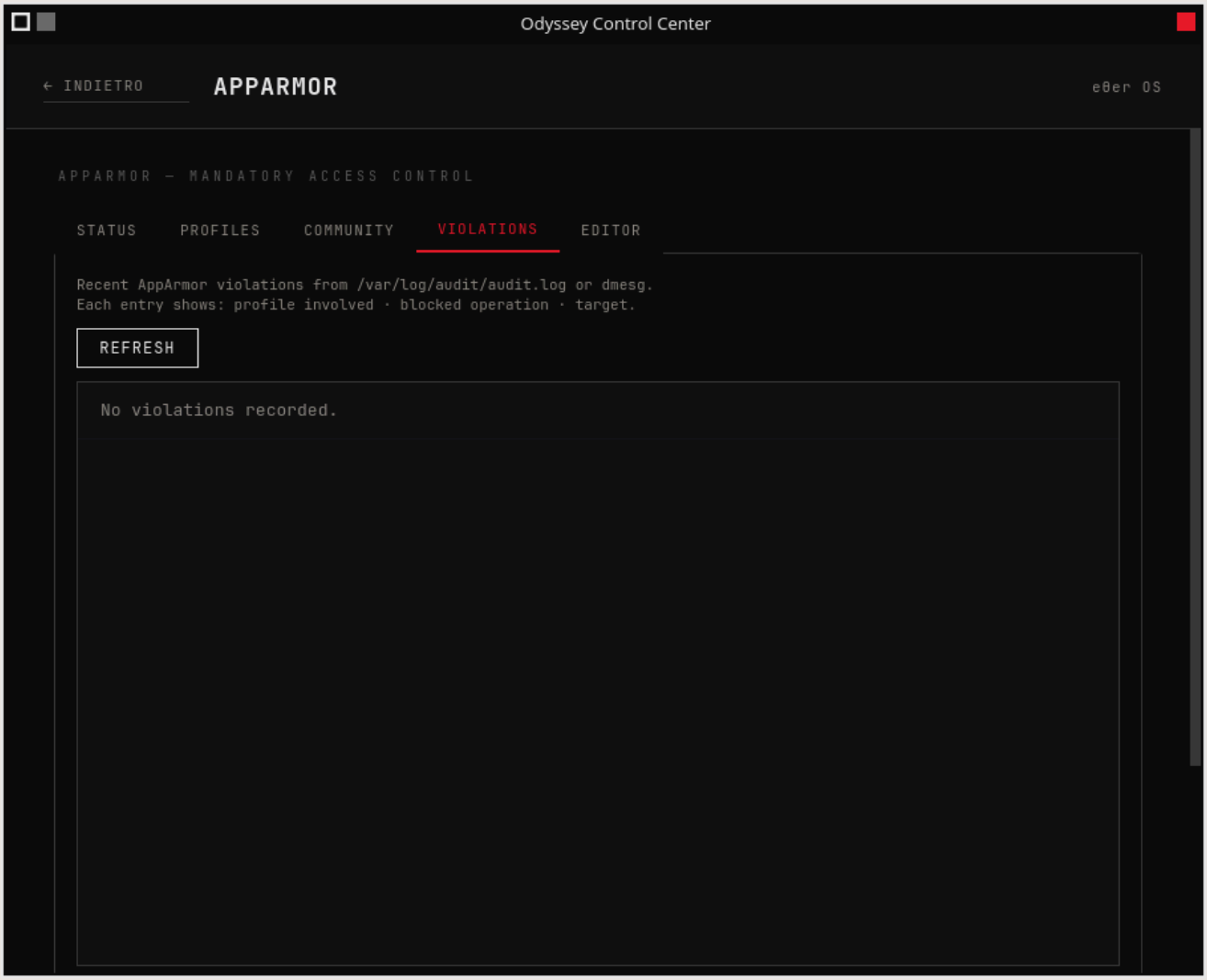

FIG · La scheda VIOLATIONS — su un sistema tranquillo e in salute, vuota è la lettura attesa. Ogni voce che compare nomina il profilo, l'operazione e il bersaglio.

VIOLATIONS legge il registro di audit al posto tuo — /var/log/audit/audit.log quando è presente un demone di audit, altrimenti il registro del kernel: ogni voce nomina il profilo, l'operazione e la risorsa che sarebbe stata negata. Fai doppio clic su una voce e l'EDITOR apre **il** profilo in questione — abbinato al nome di profilo registrato nell'evento di audit, non indovinato.

Ed ecco la funzione che sostituisce un'intera disciplina da riga di comando: il pulsante **ADD SUGGESTED RULE** dell'editor. Il pannello traduce la violazione selezionata nella regola di profilo corrispondente — una riga capability, una riga network o una riga di accesso a file — e la inserisce nel punto giusto, rispettando le regole che hai commentato. È aa-logprof, in versione grafica, una violazione alla volta, con il diff davanti agli occhi prima che tu salvi.

3.5 Il ciclo di confinamento

Tutto quanto sopra si compone in un unico flusso di lavoro ripetibile — il modo in cui AppArmor è pensato per essere usato su Odyssey:

1. **Installa** un profilo per il programma (scheda COMMUNITY, oppure uno standard già presente). Si carica in modalità complain.

2. **Convivici.** Usa il programma normalmente, con tutte le funzioni che ti servono davvero, per qualche giorno.
3. **Leggi le prove.** `VIOLATIONS` mostra che cosa il profilo avrebbe bloccato. Due verdetti possibili per voce: «questa cosa è legittima — il profilo è troppo stretto per come uso questo programma» → doppio clic, `ADD SUGGESTED RULE`, salva; oppure «il programma non ha alcun motivo di fare quella cosa» → lascia stare il profilo. Quel secondo verdetto è tutto il senso del MAC.
4. **Passa al livello superiore.** Quando il registro resta in silenzio durante l'uso normale: scheda `PROFILES`, passa a `enforce`.
5. **La ritirata è sempre aperta.** Se qualcosa si rompe sotto applicazione delle regole, riporta il profilo a `complain`, controlla la nuova violazione, aggiungi la regola, riapplica. Niente è a senso unico.

WHAT	Il ciclo qui sopra, un programma alla volta — cominciando dai programmi che affrontano input non fidato: browser, posta, chat, torrent.
WHY	Sono i programmi più esposti agli attacchi, ed esattamente quelli il cui comportamento legittimo è abbastanza stretto da poter essere confinato bene.
RESULT	Programmi che, se compromessi, restano intrappolati dentro il comportamento che hanno dichiarato. Verifica: `sudo aa-status` li elenca sotto «enforce mode».
RISK	Un profilo incompleto blocca una funzione legittima — di solito visibile come un guasto misterioso in esattamente una funzione dell'applicazione. Il ciclo è la mitigazione: prima `complain`, `enforce` solo dopo che il registro si è zittito, e un profilo alla volta così il colpevole non è mai in dubbio.

UN ESEMPIO CONCRETO: PING

Sessione reale, macchina di prova. Il profilo standard `bin.ping` con le sue regole `capability net_raw` e `network inet raw` commentate, impostato su `enforce`:

```
$  ping 172.16.1.1
ping: socktype: SOCK_RAW
ping: socket: Operation not permitted
ping: → missing cap_net_raw+p capability or setuid?
```

Questa è l'applicazione delle regole che fa il suo mestiere: `ping` ha bisogno di un socket raw, il profilo non glielo concede, il kernel dice di no. Di nuovo nel pannello: profilo su `complain`, esegui `ping` di nuovo (funziona, e registra), `VIOLATIONS` ora mostra i due dinieghi — `sendmsg`/`recvmsg` sul socket raw e la capability mancante. Doppio clic, `ADD SUGGESTED RULE` inserisce `capability net_raw,` e `network inet raw,`, salva, riapplica: `ping` confinato e funzionante. Sessanta secondi, nessun terminale, e hai capito ogni passaggio — che è lo standard a cui questo pannello si tiene.

3.6 Oltre la GUI

Le operazioni del pannello, a mano — stessi file, stesso effetto:

```
$  sudo aa-status                    # cosa è caricato, e in che modalità
$  sudo aa-enforce /etc/apparmor.d/usr.bin.example
$  sudo aa-complain /etc/apparmor.d/usr.bin.example
$  sudo apparmor_parser -r /etc/apparmor.d/usr.bin.example   # ricarica
```

Per costruire un profilo da zero quando non esiste alcun profilo della comunità, lo strumento interattivo classico è `sudo aa-genprof PROGRAMMA`: eserciti il programma in un secondo terminale, poi rispondi alle sue domande — è lo stesso ciclo osserva-poi-concedi di 3.5, guidato da un prompt invece che da un

pannello. Il risultato nasce in modalità `complain`, come tutto il resto in questo capitolo, finché il suo registro non ti convince.

LEGGERE I DINIEGHI GREZZI

La scheda `VIOLATIONS` è un lettore per il flusso di audit — `/var/log/audit/audit.log` o il registro del kernel — che puoi anche leggere direttamente:

```
$ sudo dmesg | grep -i apparmor | tail
```

Ogni evento AppArmor è una riga `audit: type=1400`. I campi che contano: `apparmor="..."` è il verdetto — `DENIED` sotto `enforce`, `ALLOWED` sotto `complain` (una violazione che **sarebbe** stata negata), `STATUS` per gli eventi di caricamento/ricaricamento dei profili, che dominano il registro e per questo scopo sono rumore. Poi `operation=` (che cosa è stato tentato — `open`, `exec`, `sendmsg`, `capable`), `profile=` (quale profilo ha deciso — il nome esatto su cui il pannello fa la corrispondenza), `name=` (il percorso o la risorsa coinvolta) e `comm=` (il processo). Verdetto, operazione, profilo, risorsa: quei quattro campi sono l'intera storia di un diniego, e sono esattamente ciò che `ADD SUGGESTED RULE` traduce in una riga di profilo.

3.7 Risoluzione dei problemi

«Ho impostato enforce e non è cambiato niente.» Se l'hai fatto fuori dal pannello: `aa-enforce` modifica i flag del profilo ma non sempre lo ricarica — esegui `apparmor_parser -r` sul profilo (il pannello lo fa per te, motivo per cui il problema compare solo nei flussi da terminale).

La scheda status mostra ambra sul controllo del kernel in esecuzione. I parametri sono in `/etc/default/grub` ma questo avvio li precede — rigenera GRUB dal pannello Boot se non l'hai fatto, poi riavvia. Fino ad allora il framework è configurato ma non attivo.

Un programma si comporta male e sospetti del suo profilo. Riporta il profilo a `complain` (non `disable`) e riprova: se il problema sparisce, `VIOLATIONS` contiene già la regola mancante, a un doppio clic di distanza. Se persiste, il profilo era innocente — e hai appena usato il MAC come strumento diagnostico, che è un uso perfettamente onorevole.

CAPITOLO 4

I default di sistema

Il firewall e AppArmor sono le difese visibili. Sotto di loro stanno i default: il kernel che Odyssey costruisce, il modo in cui i parametri regolabili persistono, come viene concesso il privilegio sul desktop e quali servizi sono in esecuzione prima che tu tocchi qualcosa. Questo capitolo li documenta tutti — inclusa, onestamente, la parte che è roadmap e non presente.

4.1 Il kernel: linux-ether

Odyssey esegue una propria build del kernel, `linux-ether`, sul ramo 6.12 LTS — una linea a supporto a lungo termine mantenuta upstream fino al 2028, che porta sopra di sé lo scheduler BORE e patch di prestazioni selezionate da CachyOS. L'LTS è di per sé una decisione di sicurezza: ogni rilascio stabile upstream su quel ramo è un flusso di correzioni verificate, consegnate a te attraverso il normale canale di aggiornamento di Odyssey.

Un passaggio dedicato di hardening della configurazione del kernel — allineato alle raccomandazioni del Kernel Self-Protection Project (KSPP): stack protector strong, operazioni di memoria fortificate, freelist SLAB irrobustite, permessi rigidi sulla memoria del kernel — è programmato per la prossima ricompilazione di `linux-ether` dopo la Launch Edition. Questa guida dichiarerà ogni opzione e il suo costo quando arriverà; fino ad allora è roadmap, e questo paragrafo ne è il resoconto onesto.

4.2 sysctl: come persiste la messa a punto

Un'installazione Odyssey appena fatta gira sui default del kernel stesso — il sistema di base deliberatamente non distribuisce alcun file di override di `sysctl`. Nel momento in cui cambi un parametro attraverso il pannello Kernel del Control Center, le tue impostazioni persistono in un unico file canonico:

```
/etc/sysctl.d/99-odyssey.conf
```

Un'eccezione per progetto: il pannello Gaming scrive i suoi parametri orientati alla sessione in `/etc/sysctl.d/999-odyssey-gaming.conf`. I drop-in di sysctl si applicano in ordine lessicale, quindi il file del gaming si ordina sempre — e quindi vince — per ultimo, senza mai litigare con le tue voci in `99-odyssey.conf`.

Tutto ciò che sta in quei file è ispezionabile e reversibile: cancella la riga, ricarica, fatto. Un insieme irrobustito di default sysctl rivolti alla rete e al kernel (`kptr_restrict`, `dmesg_restrict`, eBPF e ptrace ristretti) è nella roadmap post-lancio insieme al lavoro sul kernel qui sopra.

4.3 Privilegio sul desktop: polkit

Le azioni amministrative dagli strumenti grafici di Odyssey non ti chiedono di aprire un terminale di root. Passano da polkit — il cancello dei privilegi del desktop: un programma non privilegiato richiede **una** specifica operazione privilegiata, polkit ti autentica graficamente, e l'operazione gira con i diritti di root mentre il programma che l'ha chiesta non li ha mai.

Su Odyssey esistono due forme di questo meccanismo. Il Software Center installa le proprie policy con nome — `org.odyssey.xbps-install` e `org.odyssey.xbps-remove` — così che le operazioni sui pacchetti siano azioni dichiarate di prima classe, con le proprie regole di autorizzazione. Tutto il resto nel Control

Center passa direttamente da `pkexec`, che ha una proprietà su cui vale la pena fermarsi: la finestra di autenticazione ti mostra il **comando esatto** che sta per essere eseguito come root.

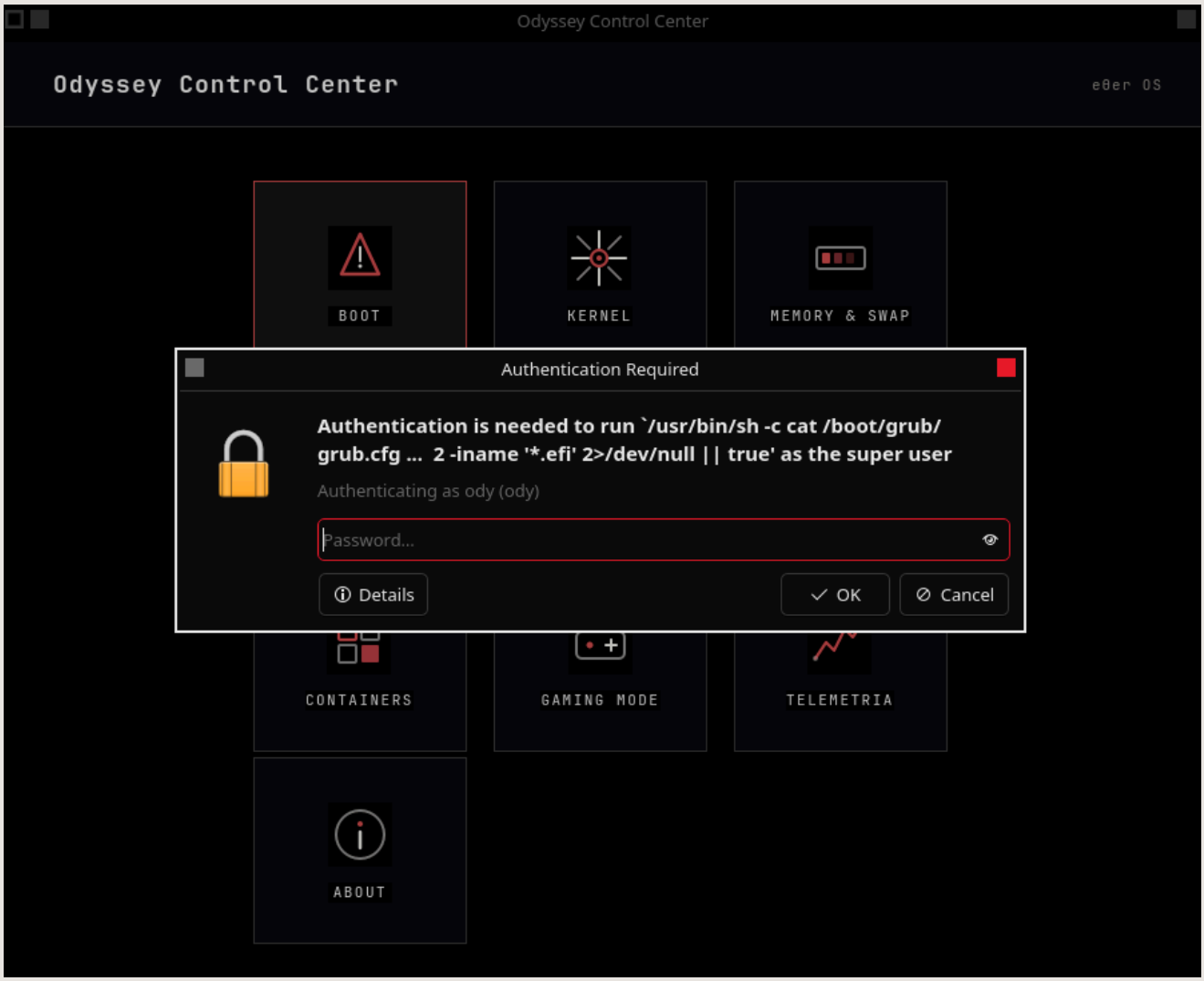

FIG · Il prompt di polkit sollevato dal Control Center. La finestra dichiara il comando esatto che viene autorizzato — non ti viene mai chiesto di firmare un assegno in bianco.

Leggere quella finestra non è paranoia; è l'abitudine che questo sistema è costruito per premiare. E dato che polkit su un sistema runit (via elogind) non mette in cache le autorizzazioni in modo affidabile tra una chiamata e l'altra, il Control Center raggruppa tutto ciò che un pannello deve fare come root in un **unico** script privilegiato — una sola richiesta di password, un ambito definito, nessuna sessione privilegiata che resta aperta. Meno richieste per progetto, mai per policy indebolita.

4.4 Servizi in esecuzione per impostazione predefinita

Odyssey pre-abilita i servizi di cui un desktop ha effettivamente bisogno — la filosofia «pronto all'uso», in deliberato contrasto con il default spoglio di Void. Ogni servizio abilitato è un collegamento simbolico in `/etc/runit/runsvdir/default/`; l'elenco autorevole sul **tuo** sistema è a un comando di distanza:

```
$ ls /etc/runit/runsvdir/default/
```

Che cosa abilita l'installer, e perché:

SERVIZIO	PERCHÉ È IN ESECUZIONE
`nftables`	Il firewall — capitolo 2. Carica il ruleset a ogni avvio.
`NetworkManager`	Connettività di rete, cablata e wireless.
`dbus`	Il bus dei messaggi del desktop; quasi tutto ciò che è grafico ne dipende.
`polkitd`	Il cancello dei privilegi della sezione precedente.
`sddm` / `lightdm` / `lxdm`	Il display manager — la schermata di accesso. Solo quello che appartiene al desktop installato esiste davvero e si avvia; gli altri nomi nell'elenco dell'installer sono inerti sul tuo sistema.
`acpid`	Eventi di alimentazione: coperchio, pulsante di accensione, batteria.
`bluetoothd`	Bluetooth, pronto senza configurazione — una scelta deliberata da sistema pronto all'uso. Se non possiedi hardware Bluetooth, è un servizio che puoi semplicemente disabilitare.
`cupsd`, `cups-browsed`	Stampa, e scoperta delle stampanti di rete.
`chronyd`	Sincronizzazione dell'ora — silenziosamente critica per la sicurezza: la validazione dei certificati TLS, e quindi ogni connessione HTTPS e ogni controllo di firma, dipende da un orologio corretto.
`sshd`	Il demone SSH — vedi sotto.
`rpcbind`, `statd`, `netmount`	Supporto NFS e filesystem di rete all'avvio.
`touchegg`	Gesti su touchpad e touchscreen.

IL CASO SSHD — DIFESA IN PROFONDITÀ, ILLUSTRATA

Sì, il demone SSH è in esecuzione per impostazione predefinita. No, la tua macchina non accetta connessioni SSH — perché la politica di scarto del firewall rende la porta 22 irraggiungibile finché non apri il preset SSH (2.6). È la stratificazione che fa il suo mestiere: lo strato dei servizi dice «disponibile», lo strato di rete dice «non raggiungibile», e abilitare l'accesso remoto è un clic deliberato invece che la configurazione di un demone. La stessa logica vale per `rpcbind` e `statd`: in ascolto localmente, invisibili dall'esterno.

Vale la pena adottare anche l'abitudine speculare: qualunque servizio della tabella che sai che non userai mai — Bluetooth su un desktop fisso, la stampa su una macchina da gioco — può essere disabilitato dal gestore dei servizi del Control Center o rimuovendone il collegamento simbolico. Un servizio che non gira non ha bug da sfruttare.

WHAT	Disabilita un servizio inutilizzato: Control Center → servizi runit, oppure `sudo rm /etc/runit/runsvdir/default/NOME` seguito da `sudo sv down NOME`.
WHY	Ogni servizio in esecuzione è superficie di attacco e memoria; l'hardening più economico è rimuovere ciò che non usi mai.
RESULT	Un demone in meno all'avvio. Verifica con `sv status NOME` e `ls /etc/runit/runsvdir/default/`.
RISK	Disabilita solo ciò che sai nominare: uccidere `dbus` o `polkitd` rompe il desktop, e uccidere `chronyd` lascia derivare l'orologio fino a far fallire il TLS. L'annullamento è lo stesso collegamento simbolico, ricreato — `ln -s /etc/sv/NOME /etc/runit/runsvdir/default/`.

4.5 Disciplina di aggiornamento

Ogni proprietà di sicurezza in questa guida decade senza aggiornamenti — il firewall protegge male un servizio non aggiornato, e un kernel LTS vale solo quanto l'ultimo incremento stabile che hai installato. Odyssey è una distribuzione rolling: gli aggiornamenti sono continui, piccoli e consegnati firmati (il capitolo 5 spiega come — e come verificarlo).

WHAT	Aggiorna regolarmente — dal Software Center o con `sudo xbps-install -Su`.
WHY	Quasi tutte le compromissioni reali di Linux desktop sfruttano vulnerabilità già corrette su macchine che non avevano applicato la correzione.
RESULT	Kernel, servizi e applicazioni allineati ai rilasci di sicurezza upstream.
RISK	Gli aggiornamenti rolling possono occasionalmente regredire; la risposta di Odyssey è il ritmo di rilascio tracciato dai bug, il gate di scansione e i pacchetti riproducibili — e la tua risposta è aggiornare spesso, così ogni aggiornamento resta piccolo e attribuibile.

CAPITOLO 5

Verificami

Ogni altro capitolo irrobustisce la tua macchina contro gli estranei. Questo irrobustisce **te** contro **noi**. Odyssey è costruito in modo che fidarsi del manutentore sia facoltativo — e questo capitolo percorre l'intera catena che lo rende possibile: ricette pubbliche, sorgenti fissati, pacchettizzazione riproducibile, una scansione antimalware che fa da cancello alla pubblicazione, due chiavi che si firmano a vicenda, un registro pubblico di trasparenza, un manifesto firmato e un cane da guardia esterno al server che avvisa **te**, non solo il manutentore. Ogni anello viene con i comandi per controllarlo, e con una dichiarazione onesta di che cosa **non** dimostra.

La versione canonica e sempre aggiornata di questo materiale vive su `odysseylinux.org/security.html`; questo capitolo ne è il compagno offline.

5.1 Il problema della fiducia

La maggior parte delle distribuzioni ti chiede di fidarti di un nome, di una fondazione, di un marchio. Odyssey non può — non ha un nome da offrire. Quindi fa il contrario: elimina del tutto la necessità di fidarsi del manutentore. Qualunque progetto può dichiarare che i suoi pacchetti sono sicuri; la dichiarazione da sola non vale nulla, perché una macchina di build compromessa, una chiave rubata o un manutentore malintenzionato producono tutti pacchetti che si installano senza problemi e sembrano normali. L'unica risposta onesta è rendere ogni anello controllabile in modo indipendente — così che una menzogna dovrebbe essere raccontata in pubblico, in più posti a prova di manomissione contemporaneamente. Un manutentore anonimo che ti lascia verificare tutto è più responsabile di uno con un nome che ti chiede di credergli sulla parola.

5.2 La catena, da un capo all'altro

Segui un pacchetto dalla sorgente al tuo disco. Ogni passo qui sotto è una barriera distinta e verificabile in modo indipendente:

1. **La ricetta è pubblica.** Ogni template di build vive su `code.odysseylinux.org/nobody` — le istruzioni, le dipendenze e le patch esatte che producono il binario.
2. **La sorgente upstream è fissata.** Il template porta con sé il checksum dell'archivio upstream; una sorgente sostituita fa fallire la build.
3. **La pacchettizzazione è riproducibile.** Ricostruire dallo stesso template dà un pacchetto identico bit per bit, così chiunque può confermare che il binario pubblicato venga davvero dalla ricetta pubblicata.
4. **Il gate di scansione.** Prima della firma, il pacchetto supera una scansione antimalware automatica (5.5). Un pacchetto segnalato non viene firmato e non entra mai nel repository.
5. **Due firme.** La chiave RSA del repository — controllata automaticamente da `xbps` a ogni installazione — e una firma cosign ECDSA pubblicata come `.cosign.bundle` accanto a ogni pacchetto.
6. **Il registro pubblico.** Ogni firma cosign è registrata in Rekor di Sigstore — un registro di trasparenza ad aggiunta sola gestito dalla Linux Foundation. Le voci sono permanenti; il manutentore non può rimuoverle né alterarle.
7. **Il manifesto firmato.** L'hash di ogni pacchetto coperto è pubblicato in un unico file firmato, `odyssey-challenge.json`, così lo stato atteso dell'intero repository è un solo documento verificabile.
8. **Il cane da guardia.** Un monitor esterno al server riverifica il repository contro quel manifesto e quelle firme ogni poche ore (5.6) — e può avvisare te, non solo il manutentore, se qualcosa fallisce.

Nessun singolo anello **è** la sicurezza. Il punto è che sconfiggere la catena significa sconfiggerla tutta, in pubblico, simultaneamente.

5.3 Verificare un pacchetto da solo

Lo strumento è `cosign` — scritto da Sigstore, usato da npm, PyPI e Kubernetes, non da Odyssey. La chiave pubblica è sulla pagina di sicurezza e nel repository `odyssey-keys`.

WHAT	Scarica un pacchetto qualsiasi e il suo bundle, poi verifica:
	`curl -LO https://repo.odysseylinux.org/odyssey-repo/x86_64/PACKAGE.xbps`
	`curl -LO https://repo.odysseylinux.org/odyssey-repo/x86_64/PACKAGE.xbps.cosign.bundle`
	
	`cosign verify-blob --key cosign.pub \\`
	`  --bundle PACKAGE.xbps.cosign.bundle PACKAGE.xbps`

WHY	`Verified OK` dimostra due cose in una volta: che il file è esattamente quello che il manutentore ha firmato, e che quella firma esiste nel registro pubblico Rekor — permanente, non rimovibile, verificabile da chiunque.
RESULT	Prova crittografica di integrità e di provenienza pubblica per i byte esatti che hai sul disco. Il bundle contiene la prova di inclusione della voce Rekor, quindi il controllo funziona anche offline; puoi anche estrarre l'indice di registro dal bundle e leggere la voce direttamente su `rekor.sigstore.dev`.
RISK	Nessuno per il tuo sistema — è in sola lettura. I limiti onesti: questo dimostra che il pacchetto è autentico e non manomesso, non che il software upstream sia privo di bug; e per i binari di terze parti ripacchettizzati, la riproducibilità convalida lo strato di pacchettizzazione di Odyssey, non il compilatore del fornitore.

Nota che cosa **non** hai dovuto fare: fidarti di questo documento, del sito web o della persona che scrive questa frase.

5.4 Un solo comando: odyssey-challenge

Per rendere il controllo indolore, il manifesto firmato `odyssey-challenge.json` pubblica l'hash atteso di ogni pacchetto coperto, e lo strumento `odyssey-challenge` verifica qualunque pacchetto installato contro quel manifesto con un solo comando. Lo stesso strumento va oltre: permette a chiunque di ricostruire un pacchetto dai sorgenti e pubblicare la propria attestazione firmata che l'hash si riproduce — conferma indipendente della dichiarazione di riproducibilità, che non poggia sulla parola di nessuno. La guida completa a verifica e ricostruzione vive su `odysseylinux.org/challenge.html`.

5.5 Il gate di scansione

Una firma dimostra che un pacchetto è autentico; non dice nulla su che cosa il pacchetto **fa**. Perciò, prima che qualunque pacchetto venga firmato, passa una scansione di sicurezza automatica che ne legge il contenuto:

Che cosa ispeziona. Corrispondenza di pattern con YARA; analisi delle capacità dei binari con capa; un inventario completo dei file installati e degli indirizzi di rete incorporati; e — la parte che conta di più

— un confronto con la versione precedente dello stesso pacchetto. Cerca accessi a credenziali e chiavi SSH, reverse shell, script di installazione che scaricano ed eseguono, miner di criptovalute, agganci di persistenza e indirizzi di comando e controllo scritti nel codice.

La difesa contro l'aggiornamento avvelenato. Il confronto tra versioni prende di mira l'attacco più difficile da individuare: un pacchetto fidato che diventa malevolo dentro un aggiornamento dall'aspetto ordinario. Se una nuova versione di routine acquisisce all'improvviso una capacità che non aveva mai avuto — leggere chiavi SSH, contattare un nuovo dominio, riscrivere il proprio script di installazione — la scansione la segnala.

Il cancello. Un pacchetto segnalato non viene firmato e non entra nel repository. Il blocco avviene prima della pubblicazione, così un pacchetto sospetto non ti raggiunge mai. Se un pacchetto segnalato appare in seguito come autorizzato, significa che un essere umano l'ha esaminato e ha verificato un falso positivo — mai il sistema che si scavalca in silenzio da solo.

Il registro è pubblico; le regole no. Ogni risultato di scansione — approvazioni, blocchi, falsi positivi autorizzati a mano — è pubblicato su `odysseylinux.org/scans.html`. Le regole e le soglie esatte sono deliberatamente riservate: pubblicare la logica di rilevamento permetterebbe a un attaccante di costruire un payload capace di scivolarci attraverso. Gli strumenti sono aperti; la loro taratura no — sicurezza operativa, dichiarata apertamente invece che nascosta.

> **CHE COSA LA SCANSIONE NON PROMETTE**
> Alza l'asticella; non è una prova di pulizia, e un impianto sofisticato e su misura può eluderla — la backdoor di XZ è scivolata oltre organizzazioni ben più grandi. È esattamente per questo che è **uno** strato nella catena di otto, mai l'intera catena.

5.6 Il cane da guardia

La crittografia dimostra che cos'è un pacchetto; non dice, da sola, a nessuno **quando** qualcosa è andato storto. Le domande che davvero ti riguardano sono: chi se ne accorge, quanto in fretta, e che cosa succede agli utenti nel frattempo.

Fuori dal server, di proposito. Il monitor non gira sulla macchina che ospita il repository — un cane da guardia su un server compromesso è un allarme che l'intruso può spegnere. Gira su infrastruttura separata, tiene la propria copia delle chiavi pubbliche e non crede a nulla di ciò che il server dei pacchetti gli dice, se non a ciò che la verifica dimostra.

Che cosa controlla, ogni poche ore. Riscarica ogni pacchetto coperto e ne verifica l'hash contro il manifesto firmato e la firma cosign contro la chiave pubblica — e controlla lo stato delle chiavi di firma stesse. Qualunque discrepanza è un fallimento, e un fallimento non è mai silenzioso: un avviso a priorità massima raggiunge ogni casella di posta del manutentore, con escalation fino a una telefonata diretta.

La risposta. Prima il contenimento: repository offline, chiavi ruotate, credenziali cambiate, firme interessate revocate; poi i pacchetti puliti rigenerati dai sorgenti. La finestra tra compromissione e risposta si misura in ore, non in settimane — il che trasforma un bersaglio già difficile in uno da cui non c'è quasi nulla da guadagnare.

Sei avvisato anche tu. Dal Software Center puoi iscriverti agli stessi avvisi: se il monitor dovesse mai segnalare una compromissione, gli iscritti ricevono la notifica via email, immediatamente — non devi seguire il progetto per sapere che qualcosa è andato storto. La vista dello stato del repository nel Software Center e la pagina di stato pubblica mostrano il verdetto di integrità corrente e quando è stato eseguito l'ultimo controllo, così puoi confermare che il sistema è sorvegliato invece di crederci sulla parola.

> **COPERTURA, ONESTAMENTE**

> Non tutti i pacchetti sono ancora nel manifesto: alcuni sono stati costruiti prima che il manifesto esistesse, quindi non c'è un hash prefirmato con cui confrontarli. Le loro ricette sono pubbliche e si ricostruiscono bit per bit come ogni altro pacchetto; man mano che vengono ricostruiti vengono aggiunti, e la pagina di stato mostra sempre la copertura dal vivo.

5.7 Le chiavi

Due chiavi, due sistemi crittografici indipendenti, un'unica catena di fiducia — e nessuna delle due legata ad alcuna identità personale:

`ether-repo` **RSA 4096.** Firma il repository dei pacchetti (i file `.sig2`); `xbps` rifiuta automaticamente qualunque pacchetto non firmato, a ogni installazione.

`cosign` **ECDSA P-256.** Firma ogni pacchetto per Sigstore e lo registra in Rekor (i file `.cosign.bundle`).

Le chiavi private vivono offline — chiavetta USB cifrata, scollegata, con una seconda copia cifrata — mai sul server, mai in un repository. Non vengono ruotate a scadenza fissa: per un progetto con un solo manutentore, la rotazione programmata aggiunge cerimonia senza sicurezza, quindi le chiavi sono a lunga durata e vengono sostituite solo se perse o compromesse — dichiarato apertamente, invece di promettere una rotazione che non avverrebbe.

Attestazione reciproca. Le due chiavi firmano incrociate un'unica dichiarazione, così ciascuna garantisce crittograficamente per l'altra. Quella singola scelta progettuale produce due garanzie: **rilevamento** — una firma che il manutentore non ha mai apposto comparirebbe in Rekor sotto gli occhi di tutti, quindi l'abuso non può restare silenzioso — e **revoca** — se una chiave viene rubata, l'altra firma un avviso di revoca autorevole che l'attaccante non può falsificare, perché avrebbe bisogno di entrambe. Una chiave rubata non può agire non vista e non può impedire la propria revoca: rilevamento verificabile della compromissione, per matematica invece che per la parola di qualcuno.

Verifica tu stesso l'attestazione — devono passare entrambi i controlli:

```
git clone https://code.odysseylinux.org/nobody/odyssey-keys
cd odyssey-keys

# 1 - firma cosign (ECDSA) + registrazione Rekor
cosign verify-blob --key cosign.pub \\
   --bundle odyssey-key-attestation.txt.cosign.bundle \\
   odyssey-key-attestation.txt

# 2 - firma ether-repo (RSA)
base64 -d odyssey-key-attestation.txt.xbps-sig.b64 > sig.bin
openssl dgst -sha256 -verify ether-repo.pub \\
   -signature sig.bin odyssey-key-attestation.txt
```

Fidati di una chiave nuova o sostitutiva solo se è annunciata sia attraverso la pagina di sicurezza sia attraverso il repository Forgejo — quel doppio canale fa parte del progetto, non è una formalità.

CAPITOLO 6

Appendice

6.1 Riepilogo dei comandi

COMANDO	CHE COSA DICE / FA
`sudo nft list ruleset`	Le regole del firewall realmente caricate nel kernel
`ss -tlnp` / `ss -ulnp`	Che cosa è in ascolto, e su quali indirizzi
`sudo nft -f /etc/nftables.conf`	Carica il ruleset persistente (in modo atomico)
`sudo sv status nftables`	Stato del servizio firewall sotto runit
`sudo aa-status`	Profili AppArmor caricati e relative modalità
`sudo aa-enforce PROFILO`	Porta un profilo all'applicazione delle regole
`sudo aa-complain PROFILO`	Riporta un profilo alla sola registrazione
`sudo apparmor_parser -r FILE`	Ricarica nel kernel un profilo modificato
`ls /etc/runit/runsvdir/default/`	Servizi abilitati all'avvio
`sudo xbps-install -Su`	Aggiornamento completo del sistema (con firme controllate)

6.2 Glossario

Drop (contro reject) — Scartare un pacchetto senza alcuna risposta. Per uno scanner, indistinguibile da un indirizzo inutilizzato.

Filtraggio stateful — Tenere traccia delle connessioni in modo che le risposte al traffico iniziato da te vengano riconosciute e lasciate rientrare.

Porta — Endpoint numerato su cui un servizio di rete è in ascolto; «aprirne» una significa permettere verso di essa il traffico in entrata non sollecitato.

MAC (controllo di accesso obbligatorio) — Restrizioni per singolo programma applicate dal kernel indipendentemente dai permessi dell'utente.

Profilo (AppArmor) — Un elenco di autorizzazione leggibile dei file e delle capability che un programma può usare.

Complain / enforce — Le due modalità di un profilo: registrare le violazioni oppure bloccarle.

polkit — Il cancello dei privilegi del desktop: policy per singola azione, autenticazione grafica, nessun root ambientale.

Registro di trasparenza (Rekor) — Un registro pubblico ad aggiunta sola delle firme; le voci non possono essere rimosse o riscritte in silenzio.

Build riproducibile — Una build che produce un artefatto identico bit per bit dagli stessi input, permettendo a terze parti di confermare la corrispondenza tra binario e ricetta.

6.3 Letture di approfondimento

`man nft` — Il riferimento completo di nftables: enorme, preciso e già installato sul tuo sistema.

Wiki di nftables — `wiki.nftables.org`, la raccolta di HOWTO upstream.

Pagine di manuale di AppArmor — `man apparmor`, `man apparmor.d` (sintassi dei profili), `man aa-genprof`.

openSUSE Security and Hardening Guide — il manuale su AppArmor più completo esistente, scritto in gran parte dagli stessi sviluppatori di AppArmor — `doc.opensuse.org`. I suoi esempi usano gli strumenti di openSUSE (YaST, zypper); la parte su AppArmor si traduce direttamente.

Documentazione di Sigstore — `docs.sigstore.dev`: cosign e Rekor, la macchina dietro il capitolo 5.

Sorgenti di Odyssey — `code.odysseylinux.org/nobody`: ogni ricetta, ogni chiave, e il sorgente di questo documento.