# Comandi per Gestione Programmi eBPF

1) Compilazione del Programma eBPF

   Compilazione con clang:

   clang -O2 -target bpf -c xdp_drop.c -o xdp_drop.o

   Compila il file sorgente xdp_drop.c per generare un file oggetto (xdp_drop.o) compatibile con eBPF.

   Compilazione con Make

   make nomeFile

   Compila il programma utilizzando il comando make, che automatizza la compilazione usando un Makefile

2) Attaccare il Programma eBPF all'Interfaccia di Rete

   sudo ip link set dev <interfaccia> xdp obj xdp_drop.o sec xdp

   Collega il programma eBPF all'interfaccia di rete specificata per iniziare a usarlo.

3) Disattivare il Programma eBPF dall'Interfaccia di Rete

   sudo ip link set dev <interfaccia> xdp off

   Scollega temporaneamente il programma eBPF dall'interfaccia di rete. Questo comando disattiva il programma eBPF senza rimuoverlo completamente.

4) Pinnare il Programma eBPF nel Filesystem BPF

   sudo bpftool prog load xdp_drop.o /sys/fs/bpf/my_xdp_prog

   Carica e pinna il programma nel filesystem BPF (/sys/fs/bpf), rendendolo persistente e accessibile per riutilizzo senza doverlo ricompilare o ricaricare.

5) Riattaccare il Programma Pinnato all'Interfaccia di Rete

   sudo ip link set dev <interfaccia> xdp pinned /sys/fs/bpf/my_xdp_prog

   Dopo aver pinnato il programma, usa questo comando per collegarlo di nuovo all'interfaccia di rete direttamente dal percorso nel filesystem BPF.

6) Rimuovere il Programma Pinnato dal Filesystem BPF

   sudo rm /sys/fs/bpf/my_xdp_prog

   Elimina il programma dal filesystem BPF quando non e' piu' necessario, liberando le risorse del kernel.

# Comandi per Verifica Programmi eBPF attivi

1) Verificare i Programmi XDP Attivi su un'Interfaccia di Rete

   sudo ip -details link show dev <interfaccia>

   Usa questo comando per vedere se c'e un programma eBPF XDP attivo su una specifica interfaccia di rete. Cerca la sezione prog/xdp nell'output per verificare la presenza di un programma attivo.

2) Verificare i Programmi Attivi sui Cgroups

   sudo bpftool cgroup tree

   Mostra l'albero dei cgroups e i programmi eBPF attivi su ciascun nodo, inclusi quelli di controllo delle risorse e di acceso ai dispositivi.

3) Verificare i Programmi eBPF Attivi su Altri Hook di Rete (non XDP)

   sudo bpftool net

   Elenca i programmi eBPF attivi su altri hook di rete, come filtri di pacchetti, NAT, o altre funzioni di rete.

4) Visualizzare Tutti i Programmi Attivi nel Kernel (Panoramica Completa)

   sudo bpftool prog show

   Mostra dettagli su tutti i programmi caricati nel kernel. Cerca dettagli come l'attributo 'attached' per individuare i programmi attivi.

## Comandi Addizionali

1) Visualizzare la lista di tutti i programmi eBPF caricati nel kernel
   sudo bpftool prog list
   Visualizza l'elenco di tutti i programmi eBPF caricati nel kernel, anche quelli non attivi.
2) Mostrare solo i processi attivi associati a terminali

   ps -a

   Visualizza solo i processi attivi associati a terminali.

   Per visualizzare senza limitazioni:
   ps -ef
   ps aux
3) Visualizzare dettagli di un programma specifico (in base a ID, nome o tag)
   sudo bpftool prog show id/name/tag <ID o nome del programma>
   Fornisce informazioni dettagliate su un programma eBPF specifico.
4) Visualizzare le informazioni dettagliate su un programma eBPF specifico

   sudo bpftool prog dump xlated name <nome_programma>

   Visualizza le istruzioni BPF "tradotte" rispetto al bytecode grezzo per un programma BPF con nome specificato, solo se il programma è attivo.

## Scrittura Programmi – Generico

1) Sezione
   SEC("xdp")
   Utilizzata per specificare il tipo di programma eBPF (ad esempio, xdp, tracepoint, ecc.) a cui collegarsi. Definisce in quale "sezione" del kernel sarà attivato il programma.
2) Stampa debug
   bpf_printk()
   Funzione utilizzata per scrivere messaggi di log o debug all'interno del kernel, simile a printf. I messaggi vengono registrati nel buffer di trace del kernel e possono essere letti con dmesg o strumenti di debug.
3) Inclusioni necessarie
   #include <linux/bpf.h>
   #include <bpf/bpf_helpers.h>
   Queste librerie forniscono le API e le definizioni essenziali per la scrittura di programmi eBPF.
4) Licenza del programma
   char LICENSE[] SEC("license") = "Dual BSD/GPL";
   Per permettere al kernel di eseguire il programma eBPF, è necessario dichiarare una licenza compatibile con i requisiti del kernel Linux. Questo permette di utilizzare tutte le API eBPF del kernel senza restrizioni.

## Scrittura Programmi – XDP

1) Struttura dei metadati
   struct xdp_md *ctx
   Puntatore a una struttura di metadati (xdp_md) che contiene informazioni sul pacchetto in arrivo. Viene passato alla funzione eBPF e fornisce accesso diretto ai dati del pacchetto.
2) Accettare e inoltrare il pacchetto
   XDP_PASS
   Accetta e inoltra il pacchetto di rete alla rete di stack di Linux. Permette al pacchetto di continuare il suo percorso normale.
3) Interrompere il processo
   XDP_ABORTED

   Interrompe il processamento del pacchetto a causa di un errore o condizione particolare. Indica un errore e interrompe il processo.
4) Scartare il pacchetto
   XDP_DROP

Scarta il pacchetto di rete, impedendogli di continuare il suo percorso. Utile per filtrare o bloccare il traffico indesiderato.

5) Inoltrare il pacchetto in loopback
   XDP_TX
   Inoltra il pacchetto direttamente sulla stessa interfaccia di rete da cui è arrivato, creando un loopback immediato senza attraversare altri livelli di rete
6) Reindirizzare il pacchetto
   XDP_REDIRECT
   Reindirizza il pacchetto verso un'altra interfaccia di rete o un target specifico (come una coda AF_XDP). Utile per bilanciare il carico o inoltrare i pacchetti a destinazioni personalizzate.
7) Accesso diretto ai pacchetti con Sockets
   AF_XDP
   è un tipo speciale di socket in Linux che permette a un'applicazione di accedere direttamente ai pacchetti di rete dalla scheda, bypassando lo stack di rete tradizionale. Ideale per applicazioni di rete ad alte performance.