eBPF thorough introduction

<https://lwn.net/Articles/740157/> -> thorough introduction

* Evoluzione
* Cosa si può fare
* Verifier
* bpf() system call
* eBPF program types
* eBPF maps
* how to write eBPF program

<https://lwn.net/Articles/742082/> -> BPF Compiler Collection introduction

<https://lwn.net/Articles/747640/> -> BCC TRACEPOINT\_PROBE

<https://lwn.net/Articles/753601/> -> eBPF tracepoint in user space

<https://docs.cilium.io/en/latest/bpf/> -> BPF and XDP reference guide

BPF è un costrutto simile a una macchina virtuale altamente flessibile ed efficiente nel kernel Linux che consente di eseguire bytecode in vari punti di hook in modo sicuro. Viene utilizzato in un certo numero di sottosistemi del kernel Linux, soprattutto in network, tracing e security (ad esempio sandboxing).

Sebbene BPF esista dal 1992, la versione estesa di Berkeley Packet Filter (eBPF) è apparsa per la prima volta nel Kernel 3.18 e rende la versione originale che oggigiorno viene definita BPF "classica" (cBPF) per lo più obsoleta.

cBPF è noto a molti come il linguaggio del filtro dei pacchetti utilizzato da tcpdump.

Al giorno d'oggi, il kernel Linux esegue solo eBPF e il bytecode cBPF caricato viene tradotto in modo trasparente in una rappresentazione eBPF nel kernel prima dell'esecuzione del programma.

Anche se il nome Berkeley Packet Filter allude a uno scopo specifico del filtraggio dei pacchetti, il set di istruzioni è abbastanza generico e flessibile in questi giorni che ci sono molti casi d'uso per BPF oltre al networking.

<https://docs.cilium.io/en/latest/bpf/resources/#bpf-users> -> Alcuni progetti che utilizzano BPF

* Tracing
  + BCC

fornire una serie di utilità di tracciamento del kernel efficienti e facili da usare, tutte basate su programmi BPF che si collegano all'infrastruttura del kernel basata su kprobes, kretprobes, tracepoint, uprobes, uretprobes e sonde USDT.

* + bpftrace

strumento di tracciamento dinamico in stile DTrace per Linux e utilizza LLVM come back-end per compilare script in bytecode BPF e utilizza BCC per interagire con l'infrastruttura di tracciamento BPF del kernel

* Altri
  + libbpf

libreria BPF generica sviluppata dalla comunità del kernel Linux come parte dell'albero dei sorgenti del kernel e consente di caricare e allegare programmi BPF dai file ELF generati da LLVM nel kernel

* + bpftool

strumento principale per l'introspezione e il debug di programmi BPF e mappe BPF e, come libbpf, è sviluppato dalla comunità del kernel Linux. Consente di scaricare tutti i programmi e le mappe BPF attivi nel sistema, scaricare e disassemblare le istruzioni BPF o JITed BPF da un programma, nonché scaricare e manipolare le mappe BPF nel sistema. bpftool supporta l'interazione con il filesystem BPF, il caricamento di vari tipi di programma da un file oggetto nel kernel e molto altro

<https://www.brendangregg.com/ebpf.html> -> eBPF tracing tools (presentations and videos)

BPF è nato come tecnologia per l'ottimizzazione dei filtri di pacchetti. Se esegui tcpdump con un'espressione (corrispondente su un host o una porta), questa viene compilata in un bytecode BPF ottimale che viene eseguito da una macchina virtuale sandbox nel kernel. Extended BPF (noto anche come eBPF) ha esteso ciò che questa macchina virtuale BPF potrebbe fare: consentirle di funzionare su eventi diversi dai pacchetti e fare azioni diverse da filtraggio.

eBPF può essere utilizzato per reti definite dal software, mitigazione DDoS (eliminazione anticipata dei pacchetti), miglioramento delle prestazioni di rete (eXpress Data Path), rilevamento delle intrusioni e altro ancora. Il suo utilizzo per gli strumenti di osservabilità viene utilizzato come mostrato nel seguente flusso di lavoro:

[Immagine che contiene diagramma, schematico

Descrizione generata automaticamente](https://www.brendangregg.com/eBPF/linux_ebpf_internals.png)

Lo strumento di osservabilità ha il codice BPF per eseguire determinate azioni: misurare la latenza, riepilogare come istogramma, acquisire le tracce dello stack, ecc. Quel codice BPF viene compilato in bytecode BPF e quindi inviato al kernel, dove un verificatore può rifiutarlo se lo è ritenuti non sicuri (che include il non consentire loop o rami all'indietro). Se il bytecode BPF viene accettato, può quindi essere collegato a diverse origini evento:

* **kprobes**: kernel dynamic tracing.
* **uprobes**: user level dynamic tracing.
* **tracepoints**: kernel static tracing.
* **perf\_events**: timed sampling and PMCs.

Il programma BPF ha due modi per ritrasmettere i dati misurati allo spazio utente: dettagli per evento o tramite una mappa BPF. Le mappe BPF possono implementare array, array associativi e istogrammi e sono adatte per il passaggio di statistiche di riepilogo.