UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Esplorazione di eBPF per Windows:
analisi dell'implementazione e confronto con Linux

Relatore:
Chiar.mo Prof. Stefano Paraboschi

Matteo Locatelli - 1059210

UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Outline

- Cos’è eBPF?
- L’ecosistema
- Il problema della portabilità
- Strumenti per sviluppare con eBPF
- Futuro di eBPF
- Conclusioni
- Riferimenti

Outline

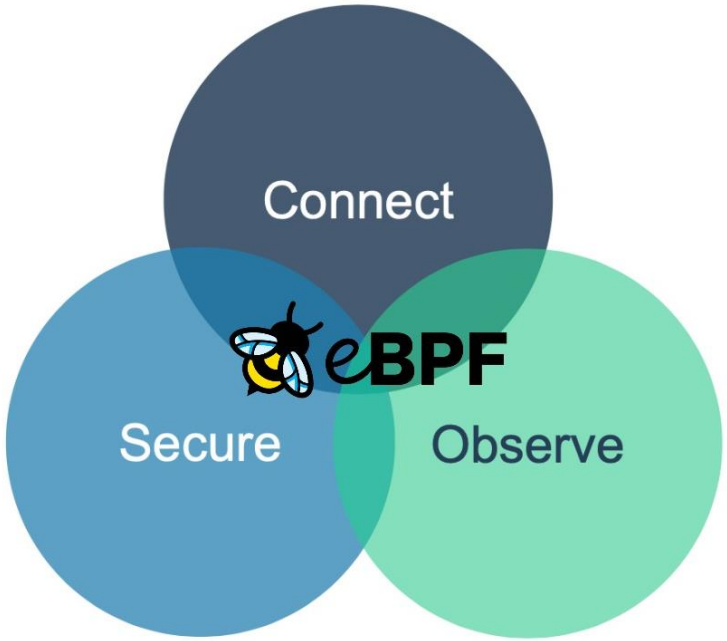
- **Cos’è eBPF?**
- L’ecosistema
- Il problema della portabilità
- Strumenti per sviluppare con eBPF
- Futuro di eBPF
- Conclusioni
- Riferimenti

eBPF: Extended Berkeley Packet Filtering

- 1992: McCanne & Jacobson pubblicano l'articolo *"The BSD Packet Filter: a New Architecture for User-Level Packet Capture"*
- Pacchetti analizzati direttamente nel kernel
- Poca flessibilità e mancanza di safety

- 7 dicembre 2014: rilascio della versione del kernel Linux 3.18
  → appare eBPF
- Porta con sé numerose novità rispetto a BPF
- Permette di eseguire programmi sandbox in un contesto privilegiato

- Oggi è il nome di una tecnologia molto usata in diversi ambiti
- Non vi è uno standard, ma è stata fondata l'«*eBPF Foundation*» e stabilito l'«*eBPF Steering Committee*»

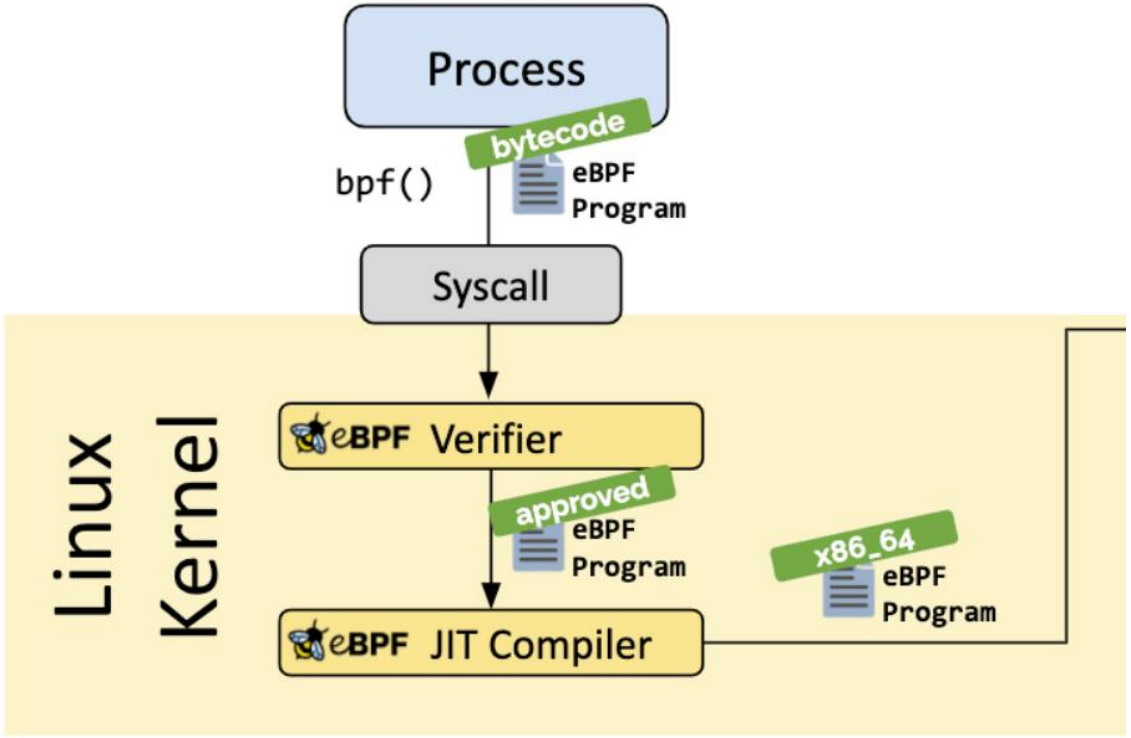


Outline

- Cos’è eBPF?
- **L’ecosistema**
- Il problema della portabilità
- Strumenti per sviluppare con eBPF
- Futuro di eBPF
- Conclusioni
- Riferimenti

Processo di esecuzione di un programma

- Scrittura del programma eBPF con un linguaggio di alto livello
- *Compilation* per generare il *bytecode* eBPF con *Clang-LLVM*
- *Verification*:
  - *Directed Acyclic Graph* check
  - Analisi di profondità del *Control Flow Graph*
- *Hardening* per aggiungere ulteriori misure di sicurezza
- *Just In Time compilation* in istruzioni macchina native
- *Loading and execution* tramite la nuova chiamata di sistema

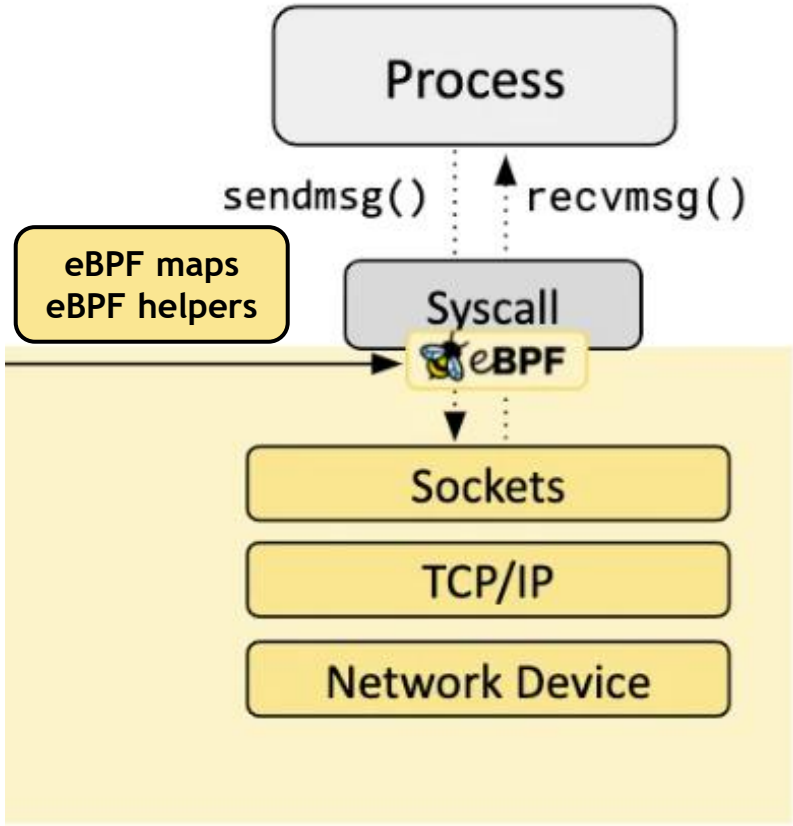


Peculiarità di un programma eBPF

- Simile ad un classico programma in C

- Svariati costrutti specifici
  - *Maps*: strutture dati per scambiare informazioni tra utente e kernel
  - *Helpers*: metodi esposti dall'API per invocare le funzioni del kernel
  - *Hook points*: punti nel kernel a cui viene attaccato il programma eBPF

- I programmi hanno memoria limitata



La chiamata di sistema bpf()

```
#include <linux/bpf.h>
int bpf(int cmd, union bpf_attr *attr, unsigned int size);
```

- *"bpf.h"*: header che contiene macro e struct che si usano con eBPF

- Interfaccia per usare le funzionalità di eBPF nel kernel
  - *"cmd"*: operazione da svolgere
  - *"attr"*: per passare dati tra utente e kernel
  - *"size"*: dimensione di *"attr"*
- Non si usa direttamente nel codice

Outline

- Cos’è eBPF?
- L’ecosistema
- **Il problema della portabilità**
- Strumenti per sviluppare con eBPF
- Futuro di eBPF
- Conclusioni
- Riferimenti

Il problema della portabilità

- "*BPF portability is the ability to write a BPF program that will successfully compile, pass kernel verification, and will work correctly across different kernel versions without the need to recompile it for each particular kernel*" (Andrii Nakryiko, 2020)

- Problema: le strutture dati all'interno del kernel cambiano tra versioni del kernel differenti

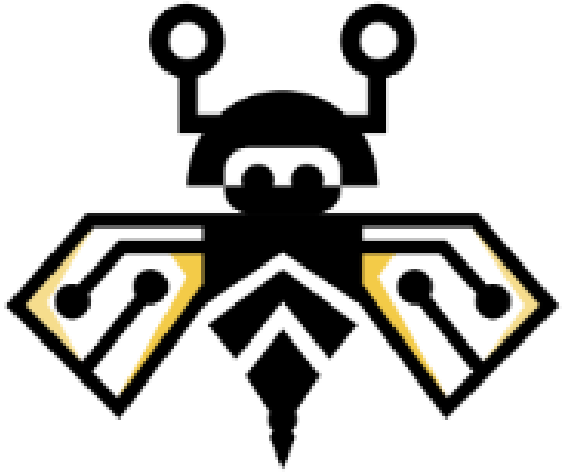
- *BPF Compiler Collection* (*BCC*): una soluzione temporanea
  - Toolkit per creare programmi eBPF
  - Diversi problemi:
    - Necessario distribuire il compilatore (Clang-LLVM) insieme all'applicazione
    - Errori rilevati solo in fase di compilazione
    - Richiede la presenza degli *header* del kernel

BPF CO-RE

- *BPF Compile Once – Run Everywhere*: la soluzione definitiva (2020)

- È l'insieme di varie novità:
  - *BPF Type Format (BTF)*: un nuovo formato dati per rappresentare le informazioni all'interno di programmi
  - Estensioni *Clang-LLVM* per accedere facilmente alle informazioni all'interno del kernel
  - Informazioni dei tipi usati dal kernel all'interno del file *"vmlinux.h"*
  - *"libbpf"*, prende il file oggetto BPF generato dopo il file processo di compilazione del programma e innesca le fasi di caricamento e verifica

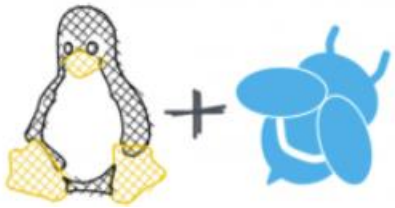
- Risultato: programma compilato ad hoc per la macchina host



Outline

- Cos’è eBPF?
- L’ecosistema
- Il problema della portabilità
- **Strumenti per sviluppare con eBPF**
- Futuro di eBPF
- Conclusioni
- Riferimenti

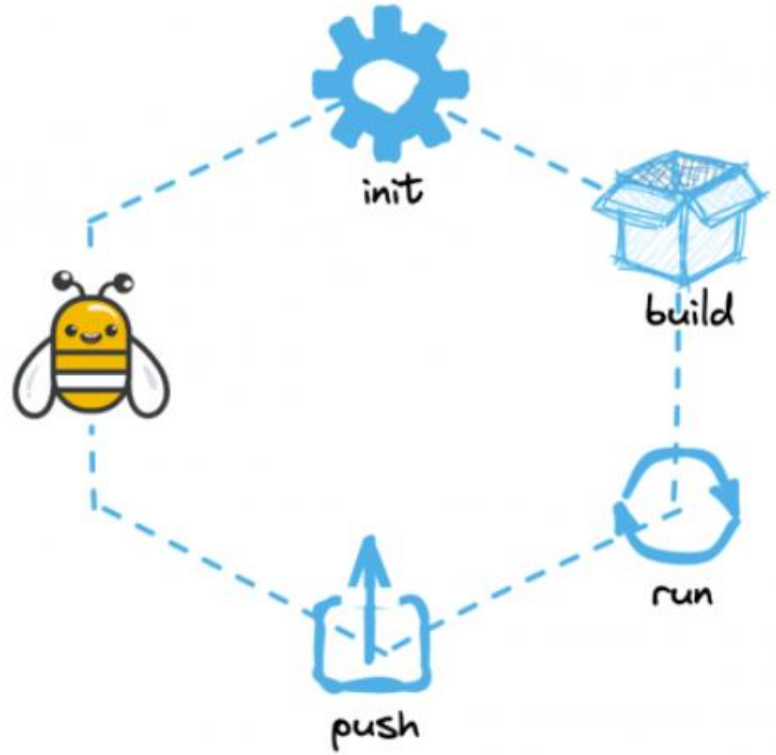
Linux: BumbleBee



- Progetto GitHub open source incentrato sulla semplificazione dell'esperienza dell'utente nella creazione di strumenti eBPF

- Sfrutta i container tramite immagini *Open Container Initiative* (*OCI*)
- Offre un'interfaccia interattiva per creare la base di un programma eBPF funzionante tramite semplici scelte

- Limiti:
  - Offre solo due hook points e due mappe
  - Mostra solo delle metriche come output
  - Gestisce in automatico il codice lato utente usando dei costrutti non tipici di eBPF



Linux: libbpf-bootstrap

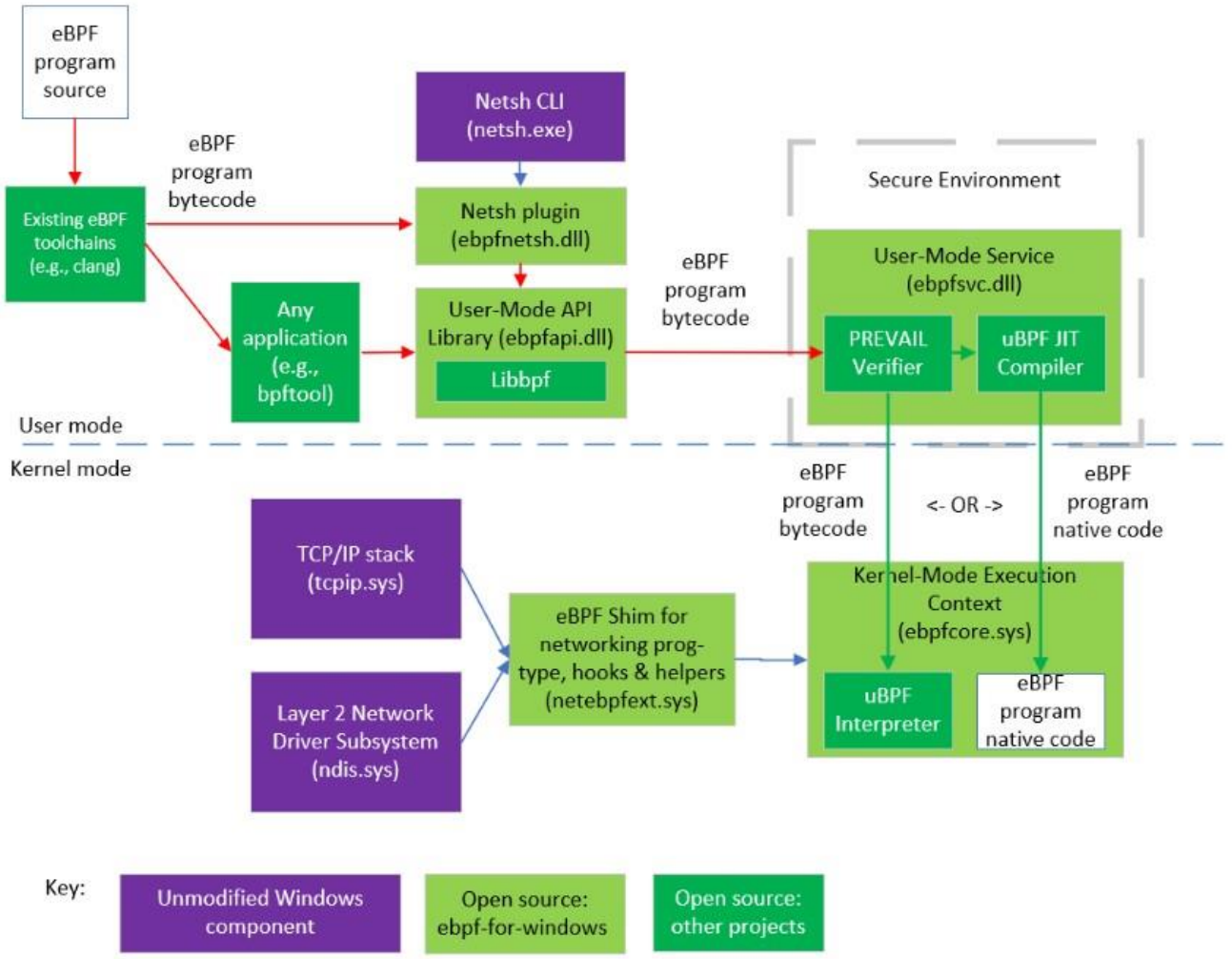
- Progetto GitHub open source basato su *libbpf*
- Permette all'utente di pensare solo al codice gestendo in autonomia le peculiarità di *libbpf*

- Sviluppo di un'applicazione in questo ambiente
  - Scrittura del programma lato kernel
  - Compilazione e generazione dello *skeleton header* e del file oggetto
  - Scrittura del codice lato utente per gestire il ciclo di vita del programma eBPF

Windows: ebpf-for-windows

- Nel maggio 2021, eBPF è stato introdotto su Windows tramite il progetto GitHub open source
- Sfrutta diversi progetti già esistenti e aggiunge le componenti per far funzionare eBPF su Windows

- Caratteristiche:
  - Esperienza dell'utente meno user-friendly
  - Numero di funzioni (helpers, hook points, …) limitate



Windows: windows-ebpf-starter

- Progetto GitHub open-source creato agli inizi del 2023 secondo la stessa idea di libbpf-bootstrap
- È un primo tentativo di creare un ambiente per sviluppare programmi eBPF su Windows

- Funzionamento più complesso
  - Non sfrutta gli *skeleton headers*
  - Alcuni header non vengono riconosciuti dal compilatore

Outline

- Cos’è eBPF?
- L’ecosistema
- Il problema della portabilità
- Strumenti per sviluppare con eBPF
- **Futuro di eBPF**
- Conclusioni
- Riferimenti

Potenziale di eBPF

- Lo stato dell'arte di eBPF su Linux e su Windows non è allo stesso punto
- Sviluppo di nuovi helpers, maps e hook points, in linea con il prevedibile progresso tecnologico

- Usi futuri
  - Osservabilità e sicurezza di sistemi e ambito network, anche in sistemi distribuiti
  - Ambito *Internet of Things* (*IoT*) (per esempio, gateways per filtraggio pacchetti in modo intelligente)
  - Migliorare la scalabilità in ambienti cloud e container
  - *Artificial Intelligence* e *Machine Learning* per decision-making intelligente e gestione delle risorse in modo dinamico
  - Nascita di nuovi frameworks prodotti dalla community

Outline

- Cos’è eBPF?
- L’ecosistema
- Il problema della portabilità
- Strumenti per sviluppare con eBPF
- Futuro di eBPF
- **Conclusioni**
- Riferimenti

"Superpowers are coming to Linux"

- Brendan Gregg, 2016

… and to Windows!

Outline

- Cos’è eBPF?
- L’ecosistema
- Il problema della portabilità
- Strumenti per sviluppare con eBPF
- Futuro di eBPF
- Conclusioni
- **Riferimenti**

Riferimenti

1. https://ebpf.io/
2. https://nakryiko.com/
3. https://docs.kernel.org/bpf/
4. https://github.com/iovisor/bcc
5. https://github.com/solo-io/bumblebee
6. https://github.com/libbpf/libbpf-bootstrap
7. https://github.com/microsoft/ebpf-for-windows
8. https://github.com/SubconsciousCompute/windows-ebpf-starter

UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Grazie per l'attenzione!

Matteo Locatelli - 1059210

UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione