Emulazione ebpf in userspace.

No in kernel windows.

Esecuzione programmi in userspace per controllare comportamento programmi.

Definire layer di traduzione e attaccata come step aggiuntivo.

Esplorazione progetto per vedere comportamento con eBPF.

Identificati limiti, pro e contro -> valutazione funzionalità da realizzare.

Per LINUX:

eBPF è una tecnologia per caricare del codice all’itnerno del kernel.

Applicazioni eseguite in user-space, ma se richiedono privilegi chiedono al kernel delle operazioni.

Kernel space: privilegiato a cui si accede tramite interfaccia in cui gira del codice a livello di root.

Codice può fare tutto.

Estensione codice -> scrittura moduli (porzioni kernel + funzionalità), mettere in sorgente, compilare e riavviare macchina,

problema: verifica compatibilità moduli con aggiornamenti

caricare programmi a livello kernel, non moduli.

Disegno macchina virtuale che emula x86 che riceve eBPF program che girano in un esecutore virtuale (processore virtuale fatto da registri su carta che esegue codice a determinati eventi).

No riscrivere kernel, ma caricare programmi e attaccare programmi a serie di eventi

* Macchina astratta che fa girare programmi attaccati ad eventi

Stessa cosa per windows

Componente esegue in kernel solo se componente di debug.

Servizio di sistema per verifier che compilazione (10 in poi)

uBPF: componente per eseguita in user-space nata con l’idea di dare messaggio chiaro

Programma scritto in sotto-set C.

Attach point (funzioni chiamate dal kernel), ricezione parametri e fare qualcosa.

Caricare programmini di attach alle funzioni e fare log su qualche struttura con lo spazio utente.

Programma kernel space -> programma in user space non accede a memoria se non viene passata.

* Studio tecnologia ed esempi su sistemi (rete e alterare flussi pacchetti con creazione proxy e firewall)

eBPF github, sito, talk.