DESCRIZIONE ESEMPIO helloworld IN libbpf-bootstrap

1. Installazione dipendenze

Installare clang, compilatore per i programmi bpf.

Clang è front-end di LLVM (in precedenza acronimo di Low Level Virtual Machine, macchina virtuale di basso livello) è un'infrastruttura di compilazione, scritta in C++, progettata per l'ottimizzazione di programmi in fase di compilazione, di linking, di esecuzione e di non utilizzo.

1. Scaricare libbpf-bootstrap

Indicazioni a <https://github.com/libbpf/libbpf-bootstrap>

libbpf-bootstrap è un semplice framework di sviluppo per lo sviluppo di applicazioni BPF basate su libbpf, dobbiamo scaricarlo.

Un framework (struttura o quadro strutturale o piattaforma software) è un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

1. Inizializzare e aggiornare le dipendenze di libbpf-bootstrap

Indicazioni a <https://github.com/libbpf/libbpf-bootstrap> (sezione CMake build)

libbpf-bootstrap ha le sue dipendenze libbpf, bpftool configurate nel suo progetto come sottomodulo git.

1. helloworld application basata sul framework libbpf-bootstrap

Nella directory “libbpf-bootstrap/examples/c” si creano due file sorgente C

* helloworld.bpf.c, il sorgente codice per il programma BPF in esecuzione nello stato del kernel
* helloworld.c, un programma in stato utente utilizzato per caricare BPF nel kernel

La logica del programma bpf in helloworld.bpf.c è iniettare bpf\_prog nel punto sepolto della chiamata execve (impostato dalla macro SEC), in modo che ogni volta che viene eseguita la chiamata execve, bpf\_prog venga richiamato.

Una **macro** è un'azione o un insieme di azioni che è possibile eseguire per un numero illimitato di volte.

**execve()** è una funzione che esegue un programma.

La logica di bpf\_prog è quella di restituire una riga di log di debug del kernel.

L'output del log si vede tramite “/sys/kernel/debug/tracing/trace\_pipe”.

Poiché il bytecode bpf è incapsulato in helloworld.skel.h, helloworld.c, che include helloworld.skel.h, è scritto in una logica più lineare: apri -> carica -> attacca -> distruggi.

Per compilare ed eseguire il programma helloworld bisogna:

* Modificare il Makefile in libbpf-bootstrap/examples/c (voce helloworld dopo la variabile APP nel Makefile)
* Eseguire all’interno della cartella libbpf-bootstrap/build i comandi “cmake ../examples/c/” e “make” per compilare le varie applicazioni tra cui helloworld.
* Eseguire helloworld con i privilegi di root (“./helloworld”)
* Eseguire con privilegi di root “cat /sys/kernel/debug/tracing/trace\_pipe” in un'altra finestra per visualizzare l'output del programma bpf (quando si verifica una chiamata di sistema execve).

Il Makefile in libbpf-bootstrap/examples/c

* Compila il bytecode bpf
* Poi lo genera in helloworld.skel.h
* Infine compila il programma helloworld che dipende da helloworld.skel.h

Eseguendo l’applicazione helloworld:

Immagine che contiene testo

Descrizione generata automaticamente

Controllando l’output in un’altra finestra dopo aver aperto alcune applicazioni (comando execve):

* nautilus-9404 per file manager
* firefox-00000, xdg-mime-00000, xdg-settings-00000 per firefox
* gnome-shell 00000
* desktop-launch 00000

al posto di 00000 c’è un numero di 4 o 5 cifre

Immagine che contiene testo

Descrizione generata automaticamente