MAPS STUDY

MAX NUMBER OF BPF\_PRINTK(“…”); ? No, il problema sta nelle istruzioni

-7: codice dopo aver fatto update usando un intero come chiave e come valore.

-17: codice dopo BPF\_NOEXIST in array map

-22 : codice di un elemento non trovato in un array map.

A volta veniva stampato, ma non so chiaramente come ottenerlo.

* Viene dato come errore quando il verificatore fallisce nel caricare e verificare il programma (di solito)

Metodi per le mappe funzionano anche se ritornano un valore “int” ma vengono usate senza assegnarle ad una variabile.

* Lookup ritorna un void\* -> puntatore a che cosa? Alla mappa? O all’elemento nella mappa?
* Update e delete ritornano un int

Update permette di assegnare il valore ritornato ad una variabile, mentre delete no

* Update non funziona con interi (example\_simple\_map)

<https://github.com/iovisor/bpf-docs/blob/master/bpf_helpers.rst>

<https://manpages.ubuntu.com/manpages/jammy/en/man7/bpf-helpers.7.html>

https://man7.org/linux/man-pages/man7/bpf-helpers.7.html

<https://prototype-kernel.readthedocs.io/en/latest/bpf/ebpf_maps.html#interacting-with-maps>

* Parametri formali
  + SEC(“…”)
  + Funzioni in .bpf.c (vuoto, void \*ctx, altro,…)
* Funzioni per le mappe
  + Lookup -> return void\*

Puntatore alla mappa o all’elemento inserito nella mappa?

* + Update e delete -> return int

Update funziona in tutti i casi

Delete funziona solo se chiamata da sola

Provando ad usarle allo stesso modo (assegnamento + richiamare variabile) delete dà errore e update no

* + Problemi con update
    - 0: uso BPF\_ANY o BPF\_EXIST

Corretto: inserimento e trova il valore

* + - -17: uso BPF\_NOEXIST

Problema: non inserisce mai una volta nemmeno all’inizio (update ritorna valore <0), ma trova sempre il valore

* + - -7: tentativo di riempire la mappa con interi (example\_simple\_map)
* Una volta che una mappa viene creata non viene più eliminata nemmeno se termina il programma?
* Dove viene creata la mappa?
* Nella documentazione si parla di un errore negativo, ma non si tratta di un semplice intero?

Bisogna trattare gli errori in un modo particolare?

SOLUZIONI (incontro 2)

Non preoccuparsi di come la memoria viene gestita dal kernel (celle sequenziali o meno)

* Delete non si può applicare a BPF\_MAP\_TYPE\_ARRAY

Ritorna un codice di errore (-22 -> -EINVAL)

* Per stampare codice di errore si deve chiamare la funzione int e fare un return 0.

Il programma capisce quando deve terminare.

* Delete si può fare in una BPF\_MAP\_TYPE\_HASH

Funzioni

* Le funzioni dentro il programma bpf devono ritornare qualcosa oppure vanni dichiarate static.

Errore: “function does not return a scalar. Only those are supported.”

Ogni cosa definita fuori da SEC è vista come globale.

Se le funzioni dichiarate fuori da SEC non sono statiche, BPF le mette in una sezione unica del programma, e possibilmente si potrebbe eccedere nella dimensione del programma massima consentita.

Ogni SEC mette a disposizione degli helpers in base al punto di attacco nel kernel

* Hooks
* Nome hook

SEC(“hook/nome\_hook\_nel\_kernel”) nome\_funzione (lista\_parametri) {…}

* Dentro la lista dei parametri ci possono essere
  + Delle funzioni definite all’interno del kernel a cui si può accedere.

Si può accedere anche ai relativi parametri per vedere cosa fa quella funzione dal punto di vista del kernel

Lista parametri -> (nome\_hook, param1, param2, …)

* + Delle strutture definite all’interno del kernel

Funzioni per mappe:

* Delete non funziona per BPF\_MAP\_TYPE\_ARRAY
* Lookup ritorna un void\* puntatore generico che punta al valore dell’elemento associato ad una determinata chiave nella mappa.

Stampando il valore si ottiene un puntatore.

Per accedere a quel valore bisogna fare un cast esplicito al tipo dei valori all’interno della mappa e dereferenziare il puntatore.