Zimný report

Jakub Bittara

Obsah

1 Popis **1**

2 Vývojové prostredie **1**

3 Vývoj **2**
3.1 Python implementácia, prechod na C implementáciu 2
3.2 CO-RE 2

4 Manuál **2**
4.1 Kompilácia 2
4.2 Prepínače 2
4.3 Použitie 3

5 Nedostatky **4**
5.1 Odstránenie dependenie na Clang/LLVM 4
5.2 Jednotný spôsob kompilácie na rôznych distribúciach 4
5.3 Problém so zapísanými blokmi 4

---

1 Popis

Výsledný program používa eBPF na zber informácii o procesoch v momente, keď sú ukončené. Tieto informácie sú napríklad vek procesu, utime, stime, uid vlastníka procesu, exit kód (viac v manuáli). Nástroj sa skladá z dvoch častí: eBPF programu, ktorý spracúva dáta z kernelu, a userspace programu, ktorý tieto dáta príjíma a interpretuje. Tento nástroj bude základom pre zber dát na analýzu procesov v letnom semestri.

2 Vývojové prostredie

Kód som testoval vo virtuálnom stroji na distribúcii Linux Mint 22, verzia kernelu 6.8.0-49-generic.

3 Vývoj

3.1 Python implementácia, prechod na C implementáciu

Prvá implementácia tohto nástroja bola v pythone s použitím knižnice bcc. Keďže tento nástroj má byť spustený na potenciálne veľkom množstve zariadení, bolo by nutné inštalovať python a ostatné dependencie na každom zariadení. Z toho dôvodu sme prešli na implementáciu v jazyku C. eBPF programy v jazyku C majú oproti tým v pythone výrazne menšiu pamäťovú stopu a menej dependencií.

3.2 CO-RE

Za účelom eliminácie nutnosti kompilovať nástroj na každom zariadení osobitne sme uvažovali nad použitím konceptu CO-RE (compile once - run everywhere). Ten by dovoľoval jednorázovú kompiláciu eBPF programu, ktorý by bežal na rôznych verziách kernelu. Tento prístup je ale možný až od verzie kernelu 5.2, čo by obmedzovalo počet zariadení, na ktorých by bolo možné nástroj spustiť, preto sme od tohto prístupu upustili.

4 Manuál

4.1 Kompilácia

Program sa kompiluje príkazom `make`.

4.2 Prepínače

- `-c | --csv`

  Výstup bude vypísaný v CSV formáte.

- `-f FILE | --file FILE`

  Výstup bude zapísaný do súbora `FILE`.

- `-F param1,param2,...,paramn | --filter param1,param2,...,paramN`

  Na výstupe sa objavia len filtrované parametre.

  Povolené hodnoty parametrov pre filter:

  - `pcomm` - názov príkazu
  - `tgid` - thread group id
  - `pid` - process id
  - `ppid` - parent process id
  - `uid` - user id majiteľa procesu
  - `age` - dĺžka života procesu v sekundách
  - `utime` - čas, ktorý proces strávil v userspace
  - `stime` - čas, ktorý proces strávil v kerneli
  - `exit` - exit kód procesu, -1 ak nedošlo k exitu

  - `exitsig` - signál, ktorý ukončil proces, -1 ak proces neskončil signálom
  - `nvcs` - počet dobrovoľných context switchov
  - `nivcs` - počet nedobrovoľných context switchov
  - `cutime` - kumulatívny `utime` dcérskych procesov
  - `cstime` - kumulatívny `stime` dcérskych procesov
  - `inblock` - počet prečítaných blokov
  - `oublock` - počet zapísaných blokov
  - `cinblock` - kumulatívny `inblock` dcérskych procesov
  - `coublock` - kumulatívny `oublock` dcérskych procesov

- `-u UID | --uid UID`

  Na výstupe sa objavia len procesy, ktoré vlastní používateľ s user id `UID`.

- `-C | --cumulative`

  Hodnoty `utime`, `stime`, `inblock`, `oublock` budú ukazovať kumulatívne hodnoty.

  Tento prepínač automaticky odfiltruje informácie o hodnotách `cutime`, `cstime`, `cinblock`, `coublock`.

- `-h | --help`

  Vypíše návod na použitie nástroja.

4.3 Použitie

Program sa spustí pomocou `make run` alebo `sudo ./main`.
Následne program beží a čaká na ukončenie nejakého procesu. Ak nejaký proces skončí, informácie o procese budú vypísané na výstup. Program je ukončený pomocou `ctrl-c` (alebo signálu `sigint`), následne vypíše počet spracovaných procesov:

```
sudo ./main --uid $(id -u) --filter pcomm,pid,age,exit
[sudo] password for user:
PCOMM           PID    age    exit
xfce4-popup-whi 4463   0.007  0
pool-spawner    4464   0.003  0
gmain           4465   0.003  0
gdbus           4466   0.003  0
gdbus           4470   0.006  0
gmain           4469   0.007  0
pool-spawner    4468   0.007  0
xfce4-popup-whi 4467   0.011  0
bash            4472   0.000  1
bash            4473   0.002  0
python3         4474   3.832  0
^C
Handled 11 processes.
```

5 Nedostatky

5.1 Odstránenie dependenie na Clang/LLVM

Momentálne sa eBPF program kompiluje cez clang. Keby sa podarilo kompilovať eBPF program pomocou gcc-bpf, odstránili by sme túto dependenciu. Tento pokus bol ale zatiaľ neúspešný.

5.2 Jednotný spôsob kompilácie na rôznych distribúciach

Niekedy je nutné kompilátoru explicitne pridať cesty k header súborom pre libbpf.

5.3 Problém so zapísanými blokmi

Jednou z informácii, ktoré program zbiera, je počet prečítaných a zapísaných blokov. Zo zatiaľ neznámych príčin program neuvádza správne hodnoty (hodnoty pre počet prečítaných a zapísaných blokov dcérskych procesov je správny). Táto nepresnosť neovplyvňuje zvyšok funkcionality nástroja.