eBPFlow

Progetto per il corso di Gestione di Reti

Alessandro Di Giorgio

Università di Pisa
A.A. 2017-2018

1 Introduzione

`eBPFlow` è un'estensione di `eBPF_TCPFlow`[1], una sonda basata su **eBPF** per la tracciatura delle chiamate di sistema `connect` e `accept` di **TCP**.

In aggiunta si propone il supporto al protocollo **UDP** e informazioni aggiuntive riguardo i flussi: container docker (se presente) e nome dell'utente che ha lanciato il processo.

In sintesi, l'architettura della sonda segue quella di `eBPF_TCPFlow`, sfrutta cioè il meccanismo delle `kprobes` per monitorare le `syscall` relative a 4 eventi:

- `TCP connect`
- `TCP accept`
- `UDP send`
- `UDP receive`

2 Informazioni aggiuntive

2.1 Username

Una delle informazioni ausiliarie che vengono fornite con i flussi è il nome utente del chiamante del processo (lo stesso che viene mostrato da `ps`).

Ottenere lo username è semplice, anche da un punto di vista computazionale. Infatti è un'informazione che è possibile prelevare direttamente dal kernel: `eBPF` mette a disposizione la funzione `bpf_get_current_uid_gid()` con cui si ottiene lo `user id`, da cui (in spazio utente, tramite funzioni di libreria) si ottiene lo username vero e proprio.

[1] https://github.com/SalvatoreCostantino/eBPF_TCPFlow

2.2 Docker

Esportare informazioni riguardo i container in cui girano i processi non è altrettanto semplice: nel kernel non c'è alcun riferimento al `container id` di cui un processo fa parte.

Le uniche informazioni fornite dal kernel (e prelevabili tramite eBPF) che si avvicinano all'identità del container sono i `namespace` (con eBPF si ricava facilmente quello dei PID) e i `cgroup`. Entrambi presentano però due problematiche diverse:

- Dai namespace non si ricava univocamente un container (e comunque bisogna passare per lo spazio utente per farlo)
- Le funzioni BPF per i cgroup sono state introdotte recentemente e sono poco supportate/documentate. In più il loro uso è limitato, ad esempio non possono essere utilizzate con le kprobes (quindi inuitile ai nostri scopi).

Per tali motivi il nome del container è ottenuto (a partire dal pid) tramite la SDK di Docker per python (quindi in spazio utente). Ovviamente questo può rallentare la sonda e portare a risultati inaccurati.

Per avviare la sonda in "modalità docker" basta passare come argomento `"-d"`. L'output sarà formattato nella colonna `COMM` come `process-name @ container-name`.

3 Supporto a UDP

L'aggiunta del supporto al protocollo **UDP** consiste in due sostanziali cambiamenti al programma originale:

- I flussi (e quindi le entry delle hash table) memorizzano anche il protocollo di livello 4 a cui il flusso fa riferimento. L'hash è calcolato anche in base a questo nuovo campo.
- Vengono installate ulteriori `kprobes` per catturare le chiamate alle funzioni `recvmsg` e `sendmsg` di UDP (nelle varianti IPv4 e IPv6).

4 Test

Viene fornito un caso di test nel file `test.sh`. Il test consiste nell'avviare un server di prova in un container, `eBPFlow` in modalità docker ed un client di prova in un altro container. Il test dovrebbe produrre un output simile al seguente:

```
Outbound flows (TCP connect, UDP send)
PID    COMM                 USER       IP PROTO SADDR                DADDR                RPORT  COUNT MTBS(s)
23087  iperf3 @ test-client root       4  TCP   172.17.0.4           172.17.0.3           5201   2     0.000110
23043  pool                 alessandro 4  TCP   192.168.1.102        216.58.198.10        443    1     -
23053  pool                 alessandro 4  TCP   192.168.1.102        216.58.198.10        443    1     -
23154  pool                 alessandro 4  TCP   192.168.1.102        216.58.198.10        443    1     -

Inbound flows (TCP accept, UDP receive)
PID    COMM                 USER       IP PROTO SADDR                DADDR                LPORT  COUNT MTBS(s)
791    systemd-resolve      systemd-re 4  UDP   192.168.1.1          192.168.1.102        46028  1     -
22975  iperf3 @ test-server root       6  TCP   ::ffff:172.17.0.4    ::ffff:172.17.0.3    5201   2     0.010787
791    systemd-resolve      systemd-re 4  UDP   0.0.0.0              127.0.0.53           13568  12    0.000886
```