Containers and BPF: twagent story

Andrey Ignatov, Facebook

October 28, 2020

twagent

- a daemon
- runs on every Facebook server
- manages all Facebook containers
- a part of the bigger TW system, see the TW paper in OSDI'20 [0]

Container (aka “task”):

- namespaces: `cgroup, mount, pid` and optionally: `ipc, net, user, uts`
- cgroup v2
- *... other usual building blocks ...*
- cgroup-bpf programs

[0] https://sites.google.com/site/tangchq/papers/Twine-OSDI20.pdf

cgroup-bpf

Vast majority of twagent tasks have one or more cgroup-bpf features enabled:

- mostly networking:
  - IP assignment (when netns is not in-use)
  - host services connector (netns is in-use)
  - transparent proxy (mostly for TLS)
  - container firewall
  - network faults injection
  - network counters (rack, datacenter, region)
- but not only:
  - sysctl access control

Let’s look at some of them ..

Example of cgroup-bpf programs (`bpftool cgroup tree`):

```
CgroupPath
ID       AttachType      AttachFlags     Name
/sys/fs/cgroup
102655   sock_ops        multi           ned_cgrp_dctcp
/sys/fs/cgroup/<path/to/task/cgroup0>
    91308    ingress         multi           tw_ingress
    91310    ingress         multi           tw_twfw_ingress
    91307    egress          multi           tw_egress
    91311    egress          multi           tw_twfw_egress
    91305    connect6                        tw_wdb_connect6
/sys/fs/cgroup/<path/to/task/cgroup1>
    80863    ingress         multi           tw_ipt_ingress
    80869    ingress         multi           tw_ingress
    80874    ingress         multi           tw_twfw_ingress
    80868    egress          multi           tw_egress
    80895    egress          multi           tw_twfw_egress
    80861    sock_ops                        tw_ipt_listen
    80859    bind6                           tw_ipt_bind
    80860    connect6                        tw_ipt_connect
    80862    sendmsg6                        tw_ipt_sendmsg
```

Task IP assignment (aka IP-per-task)

- Facebook DC network is IPv6 only
- Every server has /64 IPv6 prefix
- Convenient to have a unique IPv6 per twagent task (e.g. for QoS tagging)
- Many services don't need full L2 isolation like that of netns and don't want to pay for it
- TCP and UDP is enough

Solution:

- Make task use specified IP by a set of `BPF_PROG_TYPE_CGROUP_SOCK_ADDR` and `BPF_CGROUP_SOCK_OPS` programs

Move TCP/UDP servers to task IP:

- `bind(2): ctx.user_ip6 = task_ip`

Make TCP/UDP clients use task IP as source IP:

- `connect(2): bpf_bind(task_ip)`
- `sendmsg(2): bpf_bind(task_ip)`

Handle TCP client A connecting to TCP server B in same task by `[::1]`:

- `listen(2):` track server port by tracking `BPF_TCP_LISTEN` and `BPF_TCP_CLOSE`
- `connect(2)` to `[::1]:` redirect to `task_ip` if listener is in same task

Transparent Proxy

- Facebook traffic has to be encrypted
- Transparent TLS helps some services encrypt easily
- How to send task TCP traffic to TLS forward proxy transparently for a service?

Solution:

- Redirect client on `connect(2)` by `BPF_CGROUP_INET6_CONNECT` and `BPF_CGROUP_SOCK_OPS` programs →
- In proxy on `accept(2)` learn `orig_dst` by connection's `src` IP and port from BPF map.
- Encrypt, see [0] for details on proxy itself.

[0] https://atscaleconference.com/videos/scale-2019-enforcing-encryption-at-scale/

`BPF_CGROUP_INET6_CONNECT`:

- `orig_dst.ip = ctx->user_ip6`
- `orig_dst.port = ctx->user_port`
- Save `<socket_cookie, orig_dst>` in a map
- `ctx->user_ip6 = proxy.ip`
- `ctx->user_port = proxy.port`

`BPF_SOCK_OPS_TCP_CONNECT_CB`:

- `src.ip = ctx->local_ip6`
- `src.port = htons(ctx->local_port)`
- Replace `<socket_cookie, orig_dst>` by `<src, orig_dst>` in the map
- Garbage-collect map entry on `BPF_TCP_CLOSE` or use socket local storage for auto-cleanup

Container firewall (twfw)

- IP firewall is still useful
- Should affect only task state, not host
- Rules auto-cleanup on task stop is important
- Has to be integrated with service discovery, etc

Solution:

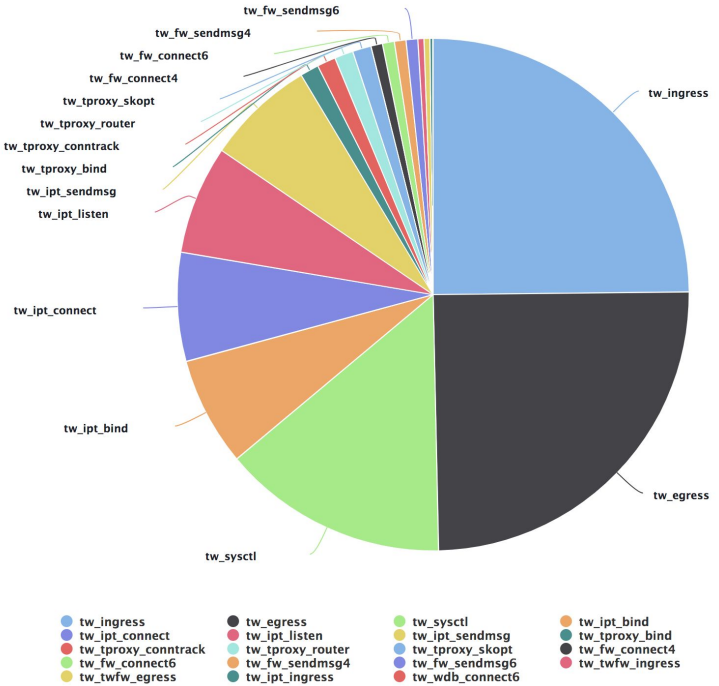
- Use `BPF_CGROUP_INET_{EGRESS,INGRESS}`
- If use-case allows, filter on socket level by `BPF_CGROUP_INET6_{CONNECT,SENDMSG}`
- Attached on task start
- Actions: pass, drop, log (via perf buffer)
- Filter by local/remote IP, IP prefix, port, protocol, TCP flags
- Integrated with service discovery: can filter by service name (dynamic set of IP:port endpoints)

Network faults injection:

- Same per-packet firewall is used
- Attached to a task on-demand by API call
- Action can be applied with probability
- Used to test disaster recovery readiness

cgroup-bpf infra

- twagent is written in C++
- `libbpf` [0] for everything-BPF
- BPF integration with `buck` [1]
- BTF [2] is enabled everywhere
- Programs and their combinations are heavily tested, incl. multi-kernel VM tests (qemu)
- Resource usage (CPU cycles, memlock) monitored across the fleet by `bpf_tax` tool →
- Alerts on program load and attach failures



[0] https://github.com/libbpf/libbpf
[1] https://buck.build/
[2] https://www.kernel.org/doc/html/latest/bpf/btf.html

Thank you!

The work presented here done by (alphabetically):

Andrey Ignatov
Dmitrii Banshchikov
Julia Kartseva
Takshak Chahande

and many others at Facebook

Q&A