ptcpdump

Process-aware, eBPF-based tcpdump

黄竹刚

中国·西安

① 目录

② 项目介绍

项目简介

ptcpdump 是一个使用 eBPF 技术开发的网络抓包工具。它的主要特点如下：

- 支持为流量关联进程、容器、Pod 信息。
- 支持对指定进程/容器/Pod 进行抓包。
- 兼容 tcpdump 常用的命令行参数以及输出格式。
- 兼容 tcpdump 使用的 pcap-filter(7) 包过滤语法。
- 静态链接，无外部依赖。

项目地址：https://github.com/mozillazg/ptcpdump

默认输出

```
$ sudo tcpdump -i any -s 0 -nnn tcp and port 80 and host www.ebpftravel.com

08:27:05.887513 ens33 Out IP 10.0.2.15.54520 > 47.114.155.73.80: Flags [S], seq 3492318931, win 64240,
options [mss 1460,sackOK,TS val 3682752063 ecr 0,nop,wscale 7], length 0
```

```
$ sudo ptcpdump -i any -s 0 -nnn tcp and port 80 and host www.ebpftravel.com

08:27:05.887454 ens33 curl.268361 Out IP 10.0.2.15.54520 > 47.114.155.73.80: Flags [S], seq 3492318931, win
64240, options [mss 1460,sackOK,TS val 3682752063 ecr 0,nop,wscale 7], length 0, ParentProc [bash.254613]
```

详细输出

```
$ sudo ptcpdump -i any -s 0 -nnn tcp and port 80 and host www.ebpftravel.com -v

18:18:15.469869 veth66f03b3c In IP (tos 0x0, ttl 64, id 49108, offset 0, flags [DF], proto TCP (6), length 60)
    10.77.0.9.57344 > 47.114.155.73.80: Flags [S], cksum 0xd53f, seq 4151361242, win 65535, options [mss 1460,sackOK,TS val 2847678216 ecr 0,nop,wscale 9], length 0
    Process (pid 11487, cmd /usr/bin/curl, args curl www.ebpftravel.com)
    User (uid 1000)
    ParentProc (pid 11348, cmd /bin/bash, args bash)
    Container (name nginx, id 5e22ae35974b39e65dad1085da9924470fbfdc14f15b268c60c124d0eaea3217, image anolis-registry.cn-zhangjiakou.cr.aliyuncs.com/openanolis/nginx:1.14.1-8.6, labels {"io.cri-containerd.kind":"container","io.kubernetes.container.name":"nginx","io.kubernetes.pod.name":"nginx-deployment-basic-59578954d4-6z9td","io.kubernetes.pod.namespace":"default","io.kubernetes.pod.uid":"4bf04b0d-dbb6-4cbd-966f-fbca5cd234f1","maintainer":"OpenAnolis Cloud Native SIG","org.opencontainers.image.created":"2022-08-29 20:11:58+0800","org.opencontainers.image.licenses":"Mulan PSL v2","org.opencontainers.image.title":"nginx","org.opencontainers.image.vendor":"Anolis OS","org.opencontainers.image.version":"1.14-8.6"})
    Pod (name nginx-deployment-basic-59578954d4-6z9td, namespace default, UID 4bf04b0d-dbb6-4cbd-966f-fbca5cd234f1, labels {"app":"nginx","pod-template-hash":"59578954d4"}, annotations {"kubernetes.io/config.seen":"2025-04-06T18:13:22.542676828+08:00","kubernetes.io/config.source":"api"})
```

PcapNG（PCAP Next Generation） 格式输出

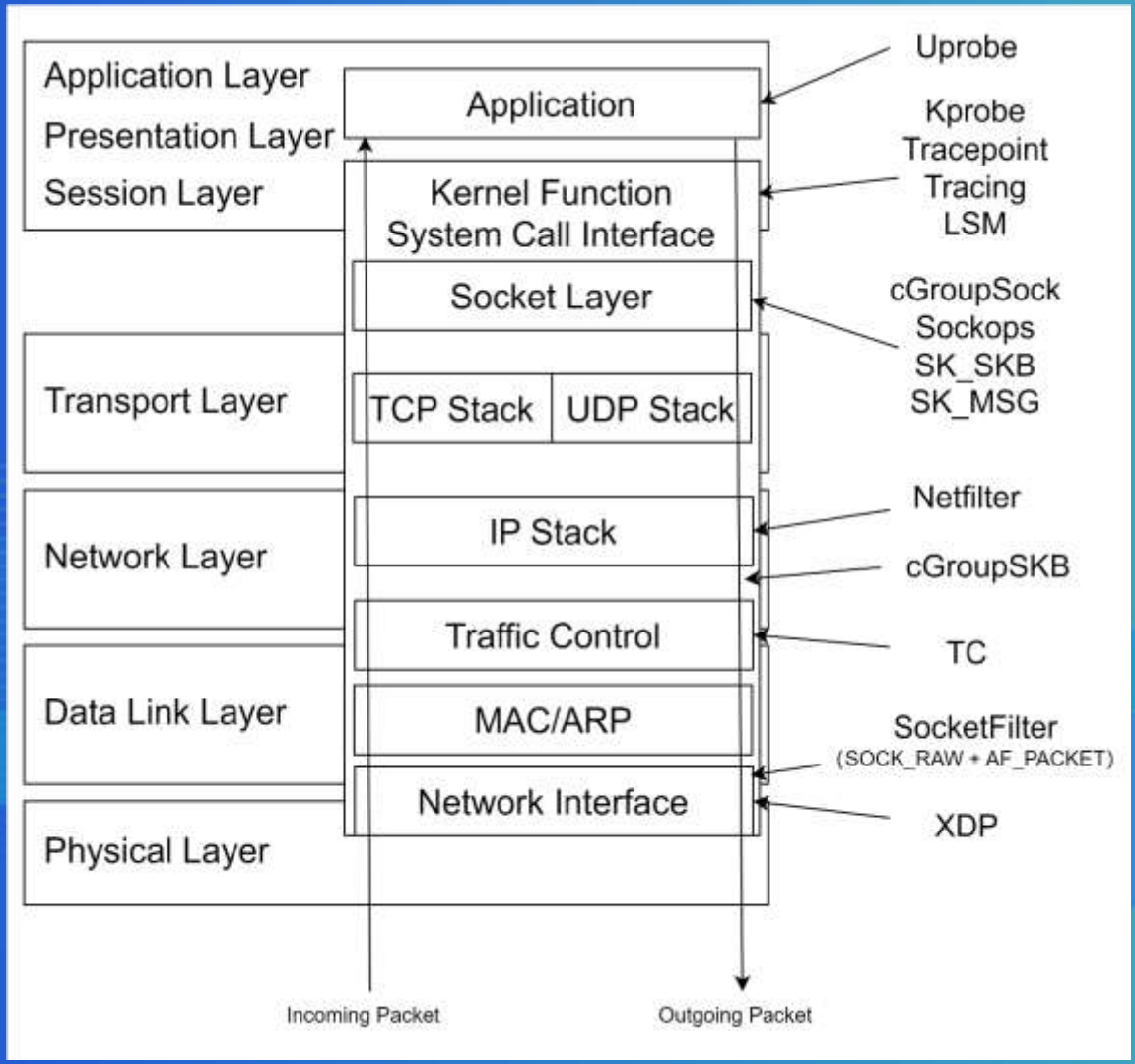
No.	Time	Source	SourcePor	Destination	DestPor	Protoco	Lengt	Info
1	2025-04-06 18:22:50.834311335	10.77.0.9	60236	47.114.155.73	80	TCP	74	60236 → 80 [SYN]

```
Packet comments
  PID: 13478 [...]
    PID: 13478
    Cmd: /usr/bin/curl
    Args: curl www.ebpftravel.com
  UserId: 1000
  ParentPID: 11348 [...]
    ParentPID: 11348
    ParentCmd: /usr/bin/bash
    ParentArgs: bash
  ContainerName: nginx [...]
    ContainerName: nginx
    ContainerId: 5e22ae35974b39e65dad1085da9924470fbfdc14f15b268c60c124d0eaea3217
    ContainerImage: anolis-registry.cn-zhangjiakou.cr.aliyuncs.com/openanolis/nginx:1.14.1-8.6
     […]ContainerLabels: {"io.cri-containerd.kind":"container","io.kubernetes.container.name":"nginx","io.kubernetes.pod.name":"nginx-deplo
  PodName: nginx-deployment-basic-59578954d4-6z9td [...]
    PodName: nginx-deployment-basic-59578954d4-6z9td
    PodNamespace: default
    PodUID: 4bf04b0d-dbb6-4cbd-966f-fbca5cd234f1
    PodLabels: {"app":"nginx","pod-template-hash":"59578954d4"}
    PodAnnotations: {"kubernetes.io/config.seen":"2025-04-06T18:13:22.542676828+08:00","kubernetes.io/config.source":"api"}
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface veth66f03b3c, id 12 (inbound)
```

③ 使用 eBPF 程序进行抓包的几种方法

中国·西安

常用的网络相关的 eBPF 程序类型



ptcpdump 使用的抓包方法

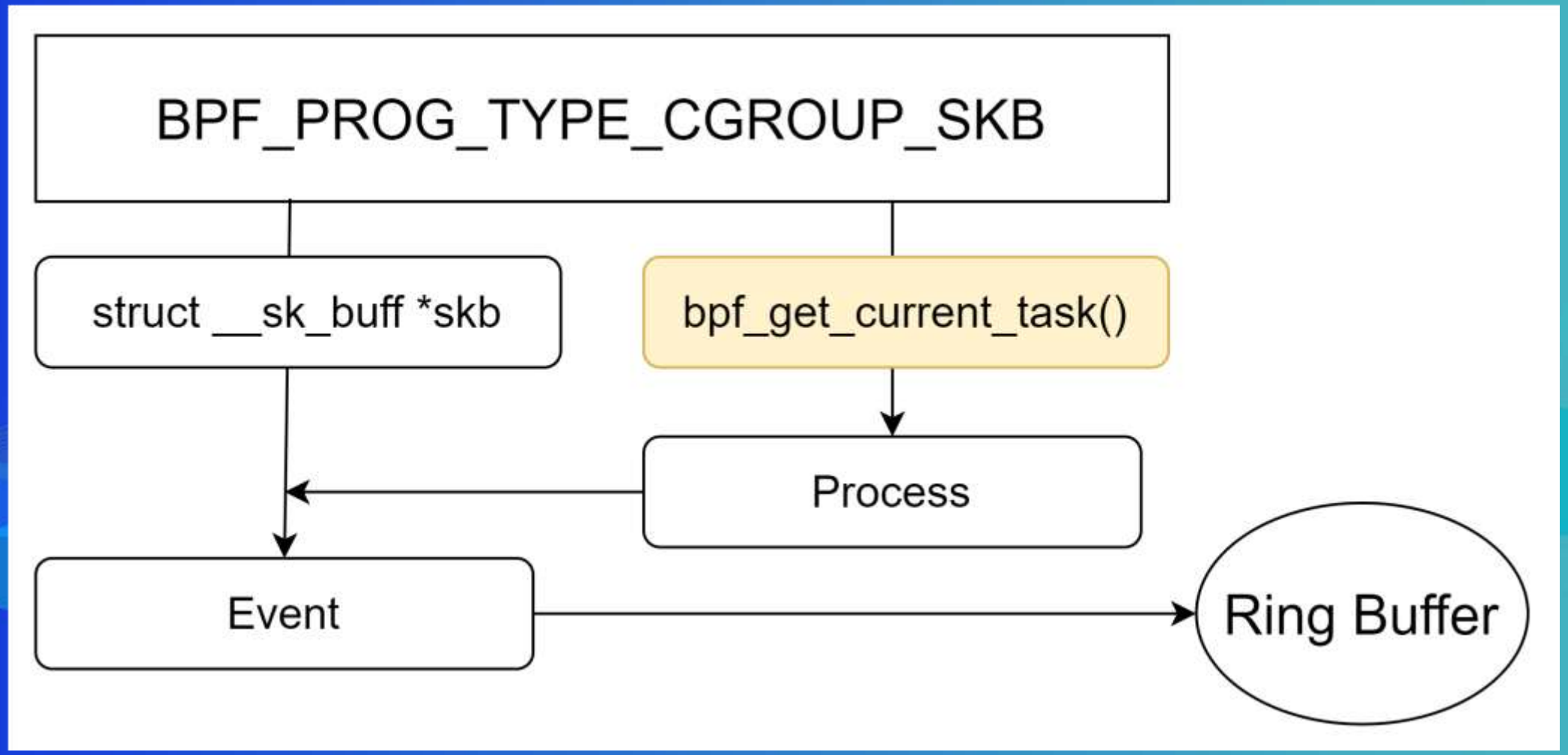
ptcpdump 当前支持通过 --backend 参数指定使用 TC 或 cGroup SKB（socket buffer）进行抓包。

--backend	eBPF Program Type	Include L2 data
`tc`	`BPF_PROG_TYPE_SCHED_CLS`	✅
`cgroup-skb`	`BPF_PROG_TYPE_CGROUP_SKB`	❌

If this flag isn't specified, it defaults to `tc`.

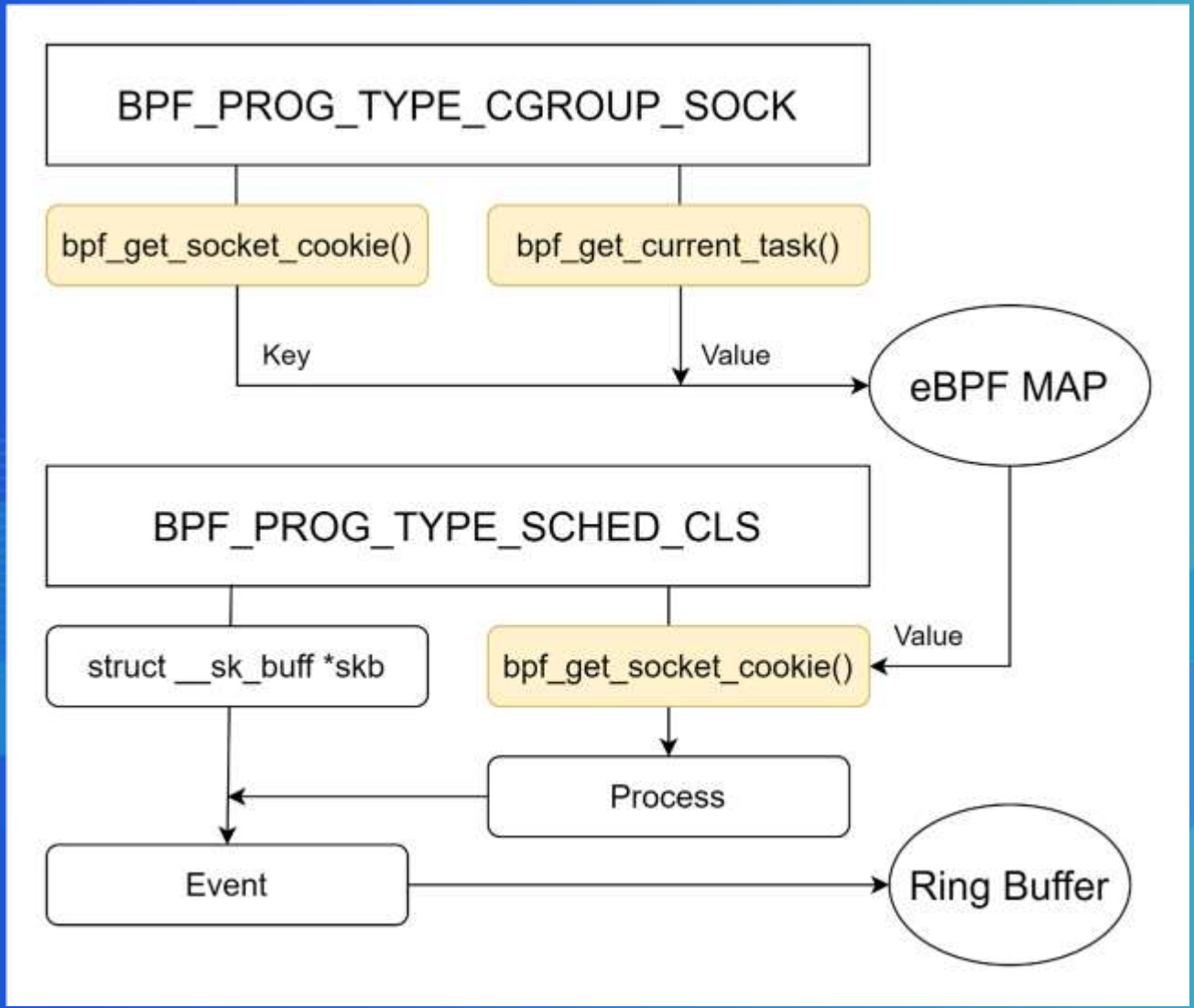
④ 为流量关联进程信息的几种方法

cGroup SKB + bpf_get_current_task()



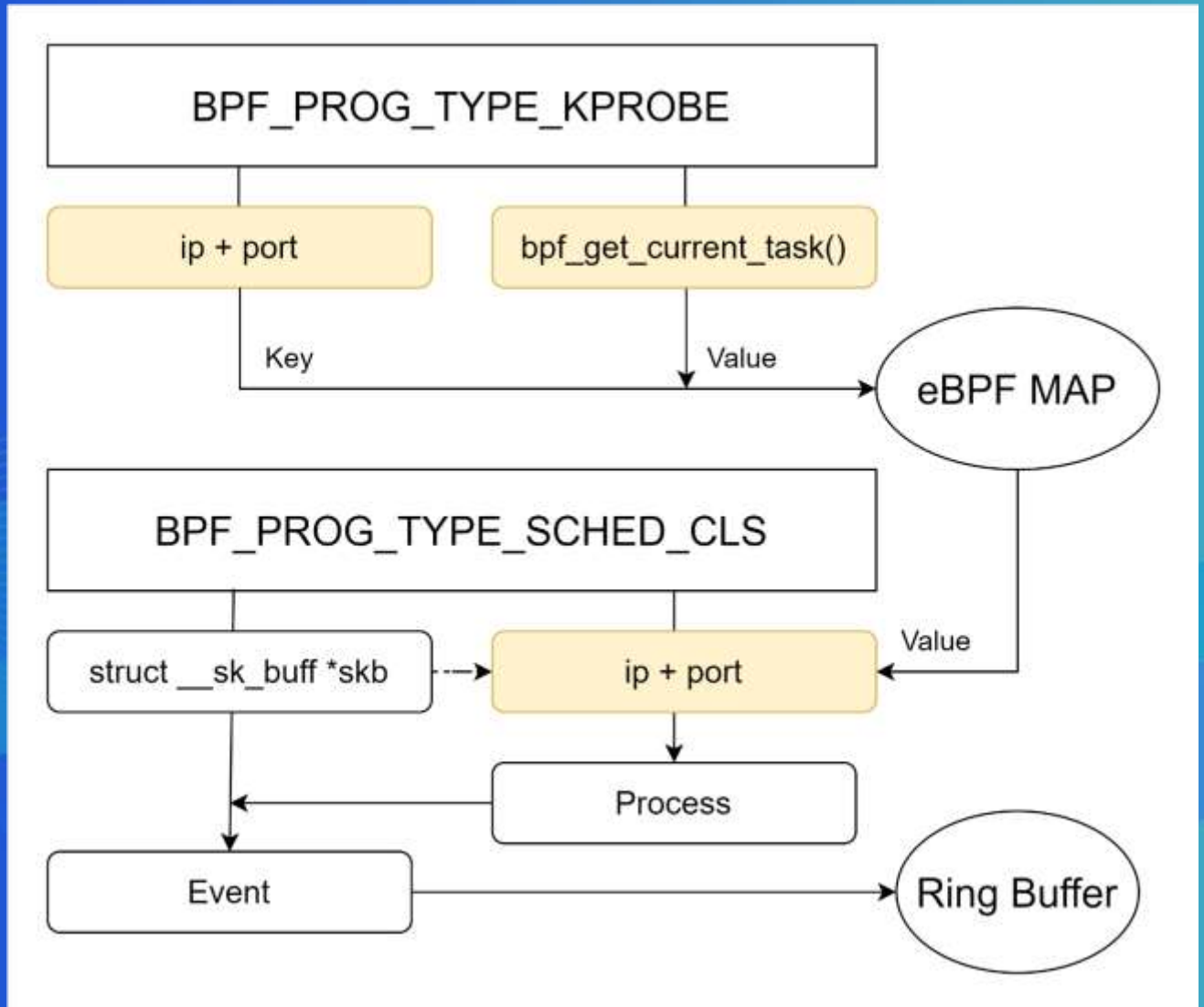
```
SEC("cgroup_skb/egress")
int cgroup_skb_egress(struct __sk_buff *skb) {
        struct task_struct *task =
          (struct task_struct *)bpf_get_current_task();
        event->process_meta = ...;
        event->data_len = ...;
        bpf_skb_load_bytes(skb, 0, &event->data, data_len);
        bpf_ringbuf_submit(event, 0);
}
```

TC + bpf_get_socket_cookie()



```
SEC("cgroup/sock_create")
int sock_create(void *ctx) {
        struct task_struct *task =
          (struct task_struct *)bpf_get_current_task();
        u64 cookie = bpf_get_socket_cookie(ctx);
        parse_task(task, &process_meta);
        bpf_map_update_elem(&sock_cookie_map, &cookie,
           &process_meta, BPF_ANY);
}
SEC("tc")
int tc(struct __sk_buff *skb) {
        u64 cookie = bpf_get_socket_cookie(skb);
        struct process_meta_t *process_meta =
           bpf_map_lookup_elem(&sock_cookie_map, &cookie);
}
```

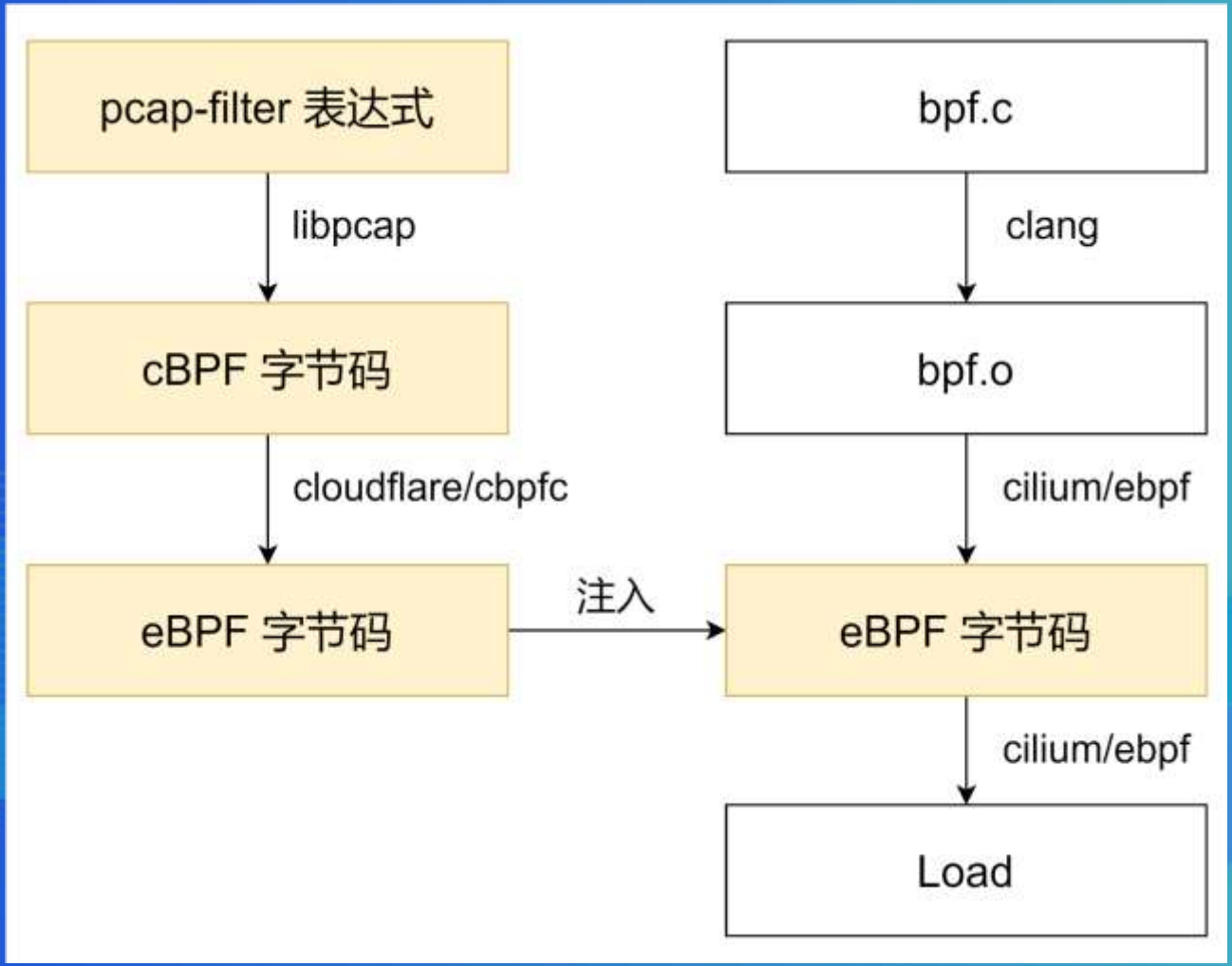
Kprobe + bpf_get_current_task()



```
SEC("kprobe/security_sk_classify_flow")
int BPF_KPROBE(security_sk_classify_flow, struct sock *sk) {
        struct task_struct *task =
          (struct task_struct *)bpf_get_current_task();
        parse_sock(sk, &flow_key);
        parse_task(task, &process_meta);
        bpf_map_update_elem(&flow_process_map, &flow_key,
          &process_meta, BPF_ANY);
}
SEC("tc")
int tc(struct __sk_buff *skb) {
        parse_skb(skb, &flow_key);
        struct process_meta_t *process_meta =
           bpf_map_lookup_elem(&flow_process_map, &flow_key);
}
```

⑤ 支持 pcap-filter (7) 包过滤语法

流程


pcap-filter 表达式
libpcap
cBPF 字节码
cloudflare/cbpfc
eBPF 字节码
注入
bpf.c
clang
bpf.o
cilium/ebpf
eBPF 字节码
cilium/ebpf
Load

示例代码

```
// r1, r2, r3, r4, r5
static __noinline bool pcap_filter(void *_skb, void *__skb,
        void *___skb, void *data, void *data_end) {
        return data != data_end && _skb == __skb && __skb == ___skb;
}
SEC("tc")
int tc_prog(struct __sk_buff *skb){
        bpf_skb_pull_data(skb, 0);
        void *data = (void *)(long)skb->data;
        void *data_end = (void *)(long)skb->data_end;
        if (!pcap_filter((void *)skb, (void *)skb, (void *)skb, data, data_end)) {
                bpf_printk("pcap_filter not match\n");
                goto out;
        }
        bpf_printk("Hello from tc after pcap filter\n");
out:
        return TC_ACT_UNSPEC;
}
```

```go
resultLabel := "result"
cbpfInsts, err := compileFilterToCbpf(expr)
if err != nil { return nil, fmt.Errorf( format: "compileFilterToCbp
ebpfInsts, err := cbpfc.ToEBPF(cbpfInsts, cbpfc.EBPFOpts{
    PacketStart: asm.R4,
    PacketEnd:   asm.R5,
    Result:      asm.R0,
    ResultLabel: resultLabel,
    Working:     [4]asm.Register{asm.R0, asm.R1, asm.R2, asm.R3},
    LabelPrefix: "pcap_filter",
})
if err != nil { return nil, fmt.Errorf( format: "ToEBPF: %w", err)
ebpfInsts = append(ebpfInsts,
    asm.Mov.Imm(asm.R1,  value: 0).WithSymbol(resultLabel),
    asm.Mov.Imm(asm.R2,  value: 0),
    asm.Mov.Imm(asm.R3,  value: 0),
    asm.Mov.Reg(asm.R4, asm.R0),
    asm.Mov.Imm(asm.R5,  value: 0),
)
```

```
$ tcpdump -d icmp
Warning: assuming Ethernet
(000) ldh      [12]
(001) jeq      #0x800          jt 2    jf 5
(002) ldb      [23]
(003) jeq      #0x1            jt 4    jf 5
(004) ret      #262144
(005) ret      #0
```

```
static __noinline bool pcap_filter(void *_skb, void *__skb,
        void *___skb, void *data, void *data_end) {
        return data != data_end && _skb == __skb && __skb == ___skb;
}
```

```
static __noinline bool pcap_filter(r1,   r2,   r3,   r4,   r5) {
        if (filter_match(r4,   r5)) {
            r0 = 262144;
        } else {
            r0 = 0;
        }
result:
        r1 = 0; r2 = 0; r3 = 0; r4 = r0; r5 = 0;
        return r4 != r5 && r1 == r2 && r2 == r3;
}
```

https://github.com/mozillazg/ebpf-program-support-pcap-filter-demo

github.com/jschwinger233/elibpcap

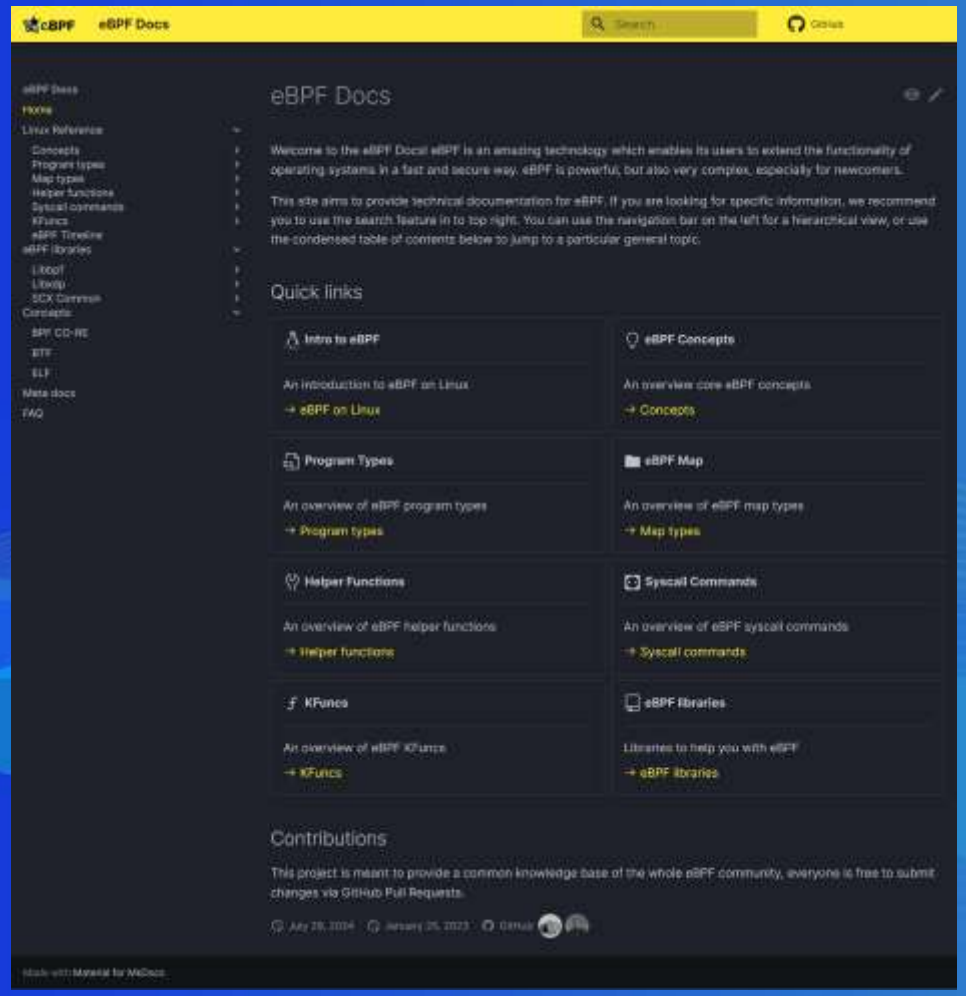
一行代码实现将 pcap-filter 表达式转换为 eBPF 字节码并注入到原有 eBPF 字节码中：

```
newInsts, err := elibpcap.Inject(expr, oldInsts, elibpcap.Options{
    AtBpf2Bpf:  "pcap_filter",
    DirectRead: true,
    L2Skb:      true,
})
```

⑥

资源推荐

docs.ebpf.io



Slack#ebpf

Telegram 频道

@W2TBP

t.me/w2tbp

梁之川梁老师是一个问题解决者，擅长解决各种疑难杂症，能解常人所不能解的 BUG，梁老师是我膜拜和仰望的大牛。这个频道时常会分享各种 eBPF 和 DEBUG 技巧以及新颖的开创性想法。

@EBPFTALKER

t.me/eBPFTalker

人称“卷王”的黄富黄老师每天都在给 eBPF 社区做贡献，不是在给内核提交 eBPF 补丁，就是在开发既实用又强大的 eBPF 应用。这个频道时常会分享各种 eBPF 知识以及老师的新项目、新贡献。

谢谢！