第二届 eBPF开发者大会

www.ebpftravel.com

基于eBPF的下一代网络抓包工具

周锋
字节跳动STE团队

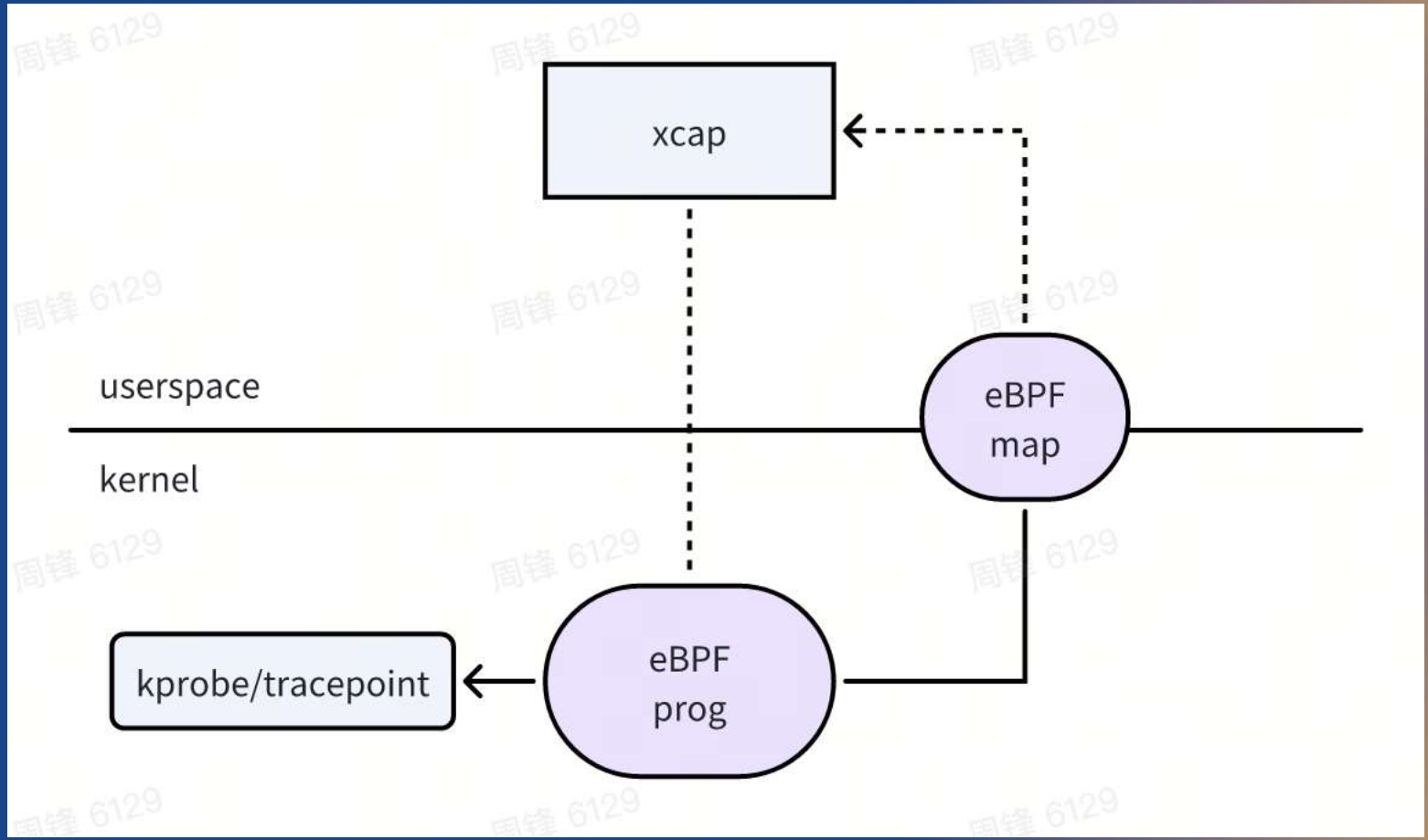
- 简介
- 典型使用场景
- 实现原理
- 未来计划
- Q&A

简介

- tcpdump抓包点位置固定:入向是xdp之后，tc之前；出向是tc之后
- bpftrace+skboutput无法做到tcpdump语法进行过滤

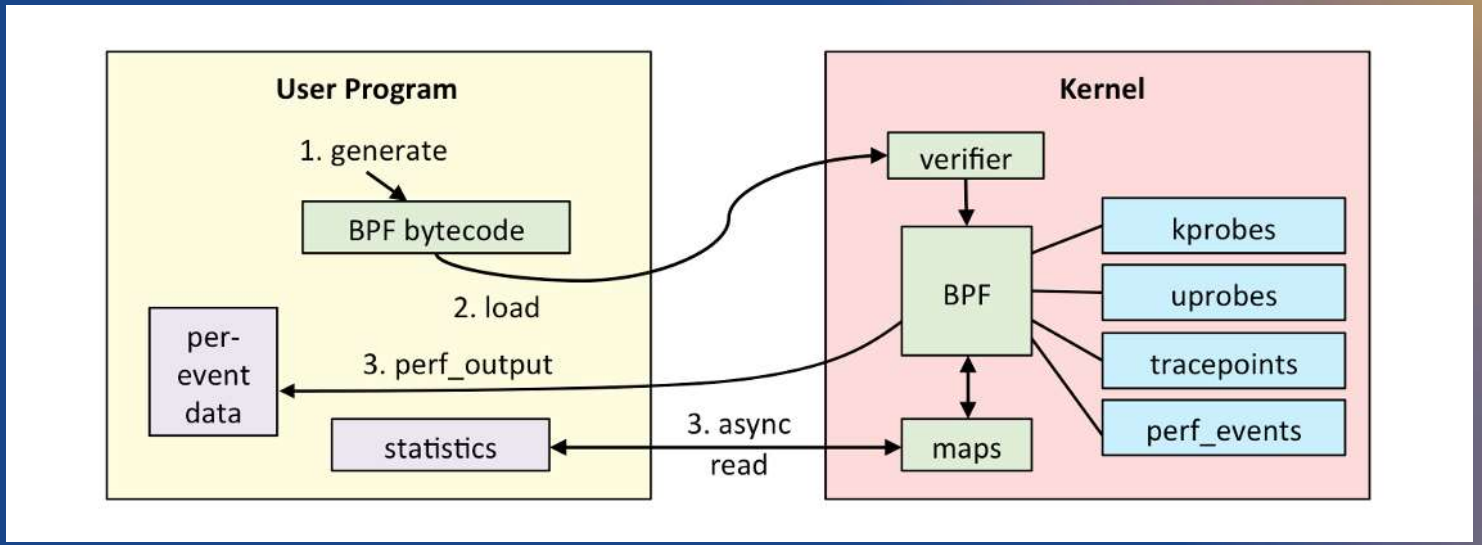
简介

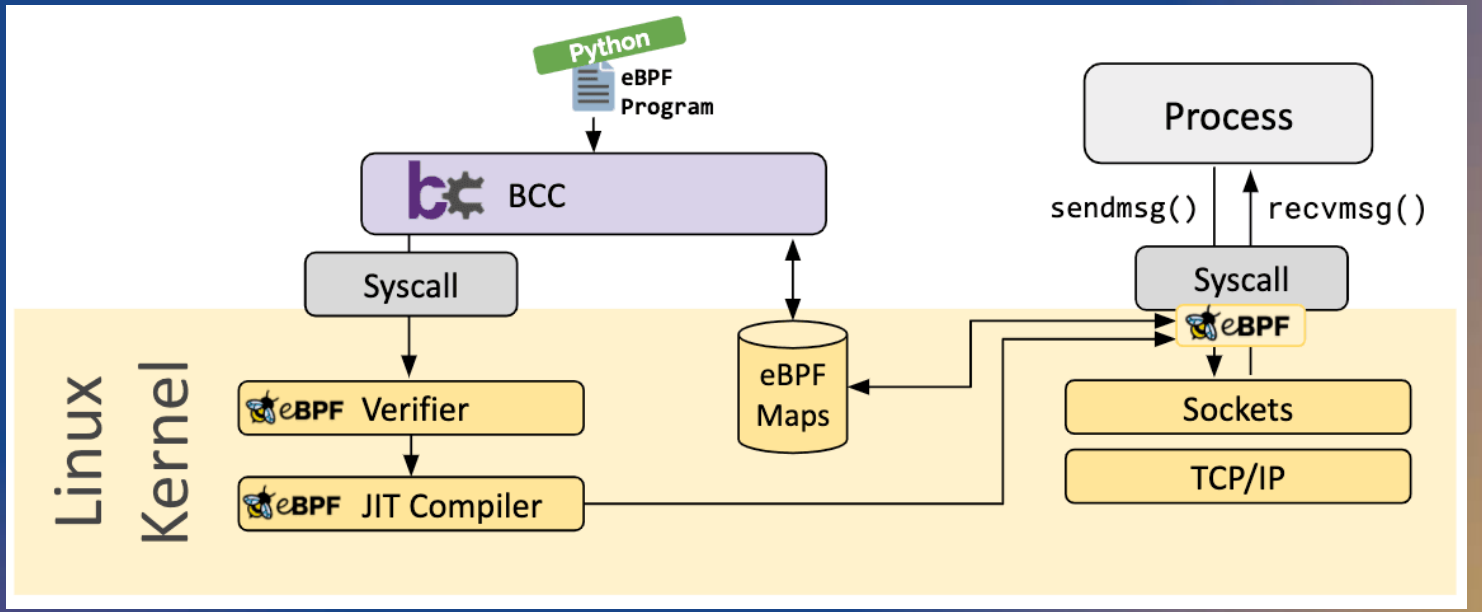
- 自定义抓包位置
- 功能可高度定制



简介

- 基于eBPF进行hook，保证安全和灵活
- 依赖bcc生成字节码，并load和attach





典型应用场景

1. 内核丢包

```
# ip -o a show dev eth1
3: eth1    inet 192.168.1.20/24 scope global eth1\      valid_lft forever preferred_lft forever

# ping 192.168.1.19
```

vm20
vm19 ⋮ eth1

```
# iptables -I INPUT -i eth1 -p icmp -j DROP
# xcap backtrace -f kfree_skb_reason -a 1 -e "icmp and dst host 192.168.1.19"
^C
[
  kfree_skb_reason
  nf_hook_slow
  ip_local_deliver
  ip_sublist_rcv_finish
  ip_sublist_rcv
  ip_list_rcv
  __netif_receive_skb_list_core
  netif_receive_skb_list_internal
  gro_normal_list.part.150
  napi_complete_done
  XXXX_napi_poll    [XXXX]
  __napi_poll
  net_rx_action
  __softirqentry_text_start
  irq_exit_rcu
  common_interrupt
  asm_common_interrupt
  native_safe_halt
  acpi_idle_do_entry
  acpi_idle_enter
  cpuidle_enter_state
  cpuidle_enter
  do_idle
  cpu_startup_entry
  start_secondary
  secondary_startup_64_no_verify
]: 3

# xcap dump -f kfree_skb_reason -a 1 -e "icmp and dst host 192.168.1.19"
19:45:12.844325 IP 192.168.1.20 > 192.168.1.19: ICMP echo request, id 40676, seq 15, length 64
19:45:13.868237 IP 192.168.1.20 > 192.168.1.19: ICMP echo request, id 40676, seq 16, length 64
19:45:14.893168 IP 192.168.1.20 > 192.168.1.19: ICMP echo request, id 40676, seq 17, length 64
```

典型应用场景

2. AF_XDP环境抓包

```
# xcap backtrace -f __dev_direct_xmit -a 1 -e "udp and dst port 4096"
^C
[
  __dev_direct_xmit
  __xsk_sendmsg
  sock_sendmsg
  __sys_sendto
  __x64_sys_sendto
  do_syscall_64
  entry_SYSCALL_64_after_hwframe
]: 10
```

```
# tcpdump -i eth1 -nn udp and dst port 4096
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
```

```
# xcap dump -f __dev_direct_xmit -a 1 -e "udp and dst port 4096"
20:24:23.765655 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765665 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765666 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765668 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765669 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765670 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765672 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765673 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765674 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:24:23.765675 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
```

```
bpf-examples/AF_XDP-example# ./xdpsock -i eth1 -t -C 10 -Q
.....
 sock0@eth1:0 txonly xdp-drv
                pps         pkts        2.00
rx              0           0
tx              5           10
```

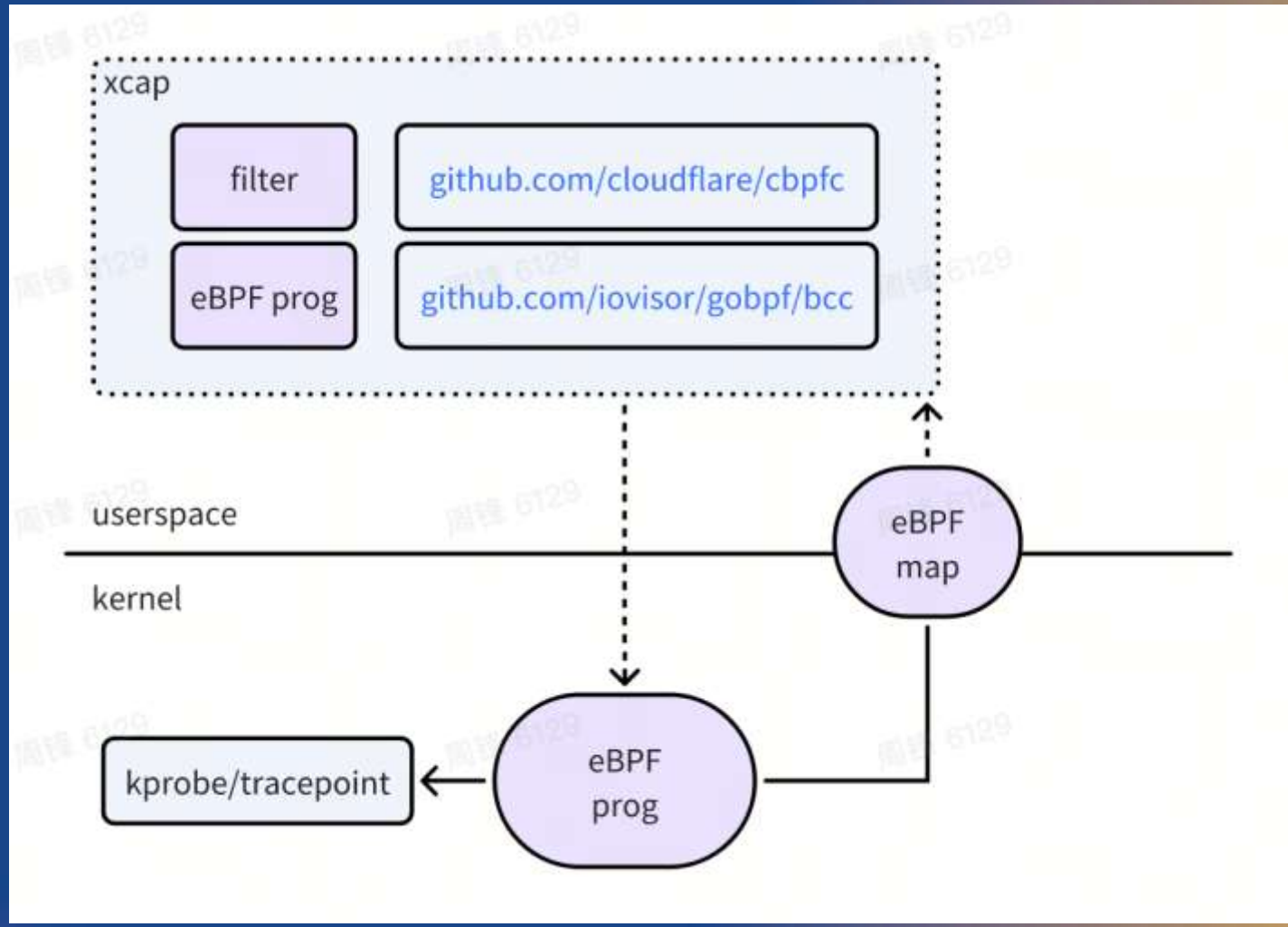
vm20

vm19 — eth1

```
# tcpdump -i eth1 -nn udp and dst port 4096
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
20:30:48.075894 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:30:48.075897 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:30:48.075898 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:30:48.075898 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:30:48.075899 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:30:48.075899 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
20:30:48.075900 IP 10.10.10.16.4096 > 10.10.10.32.4096: UDP, length 18
^C
7 packets captured
10 packets received by filter
3 packets dropped by kernel
```

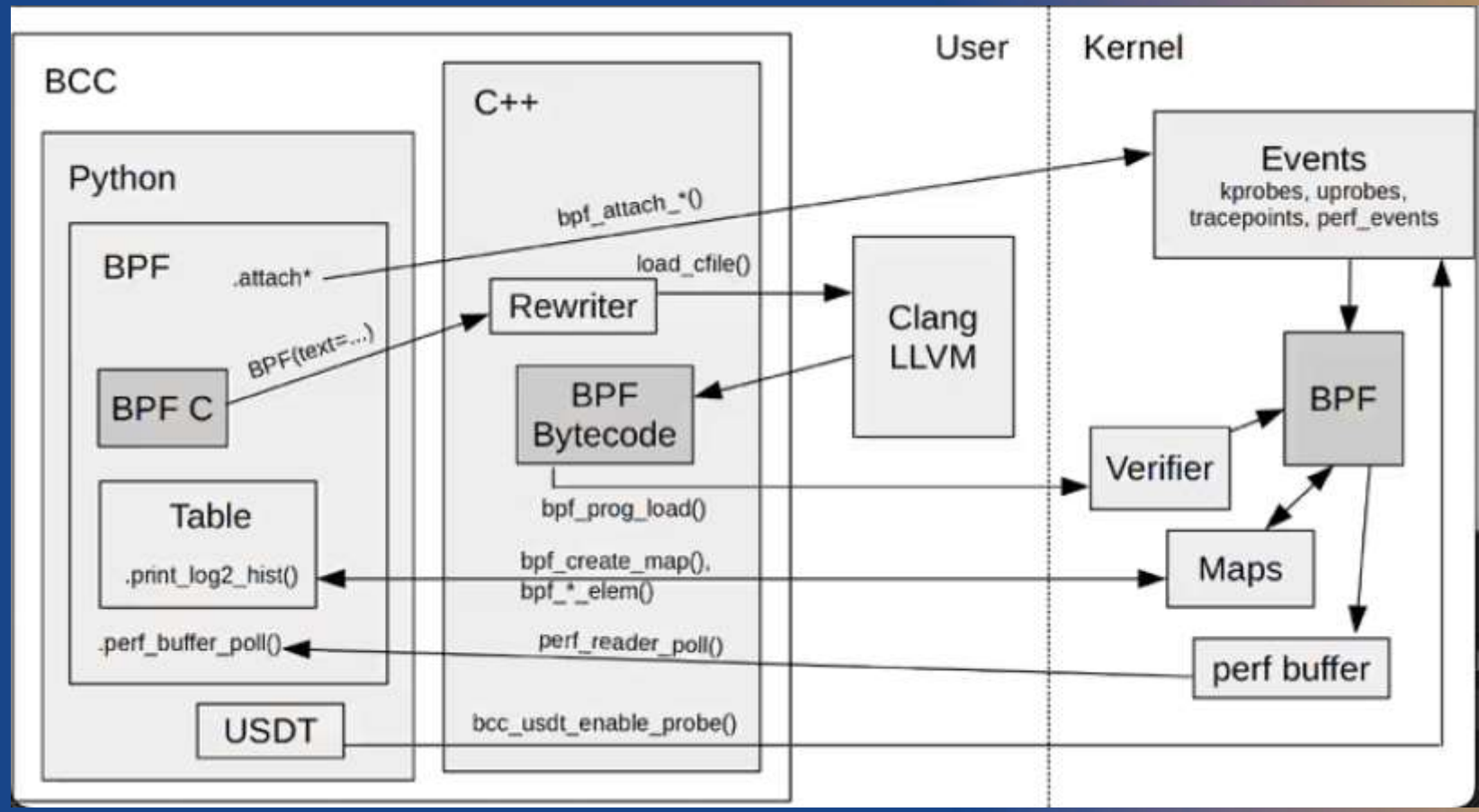
实现原理

整体架构图


xcap
filter
github.com/cloudflare/cbpfc
eBPF prog
github.com/iovisor/gobpf/bcc
userspace
kernel
eBPF
map
kprobe/tracepoint
eBPF
prog

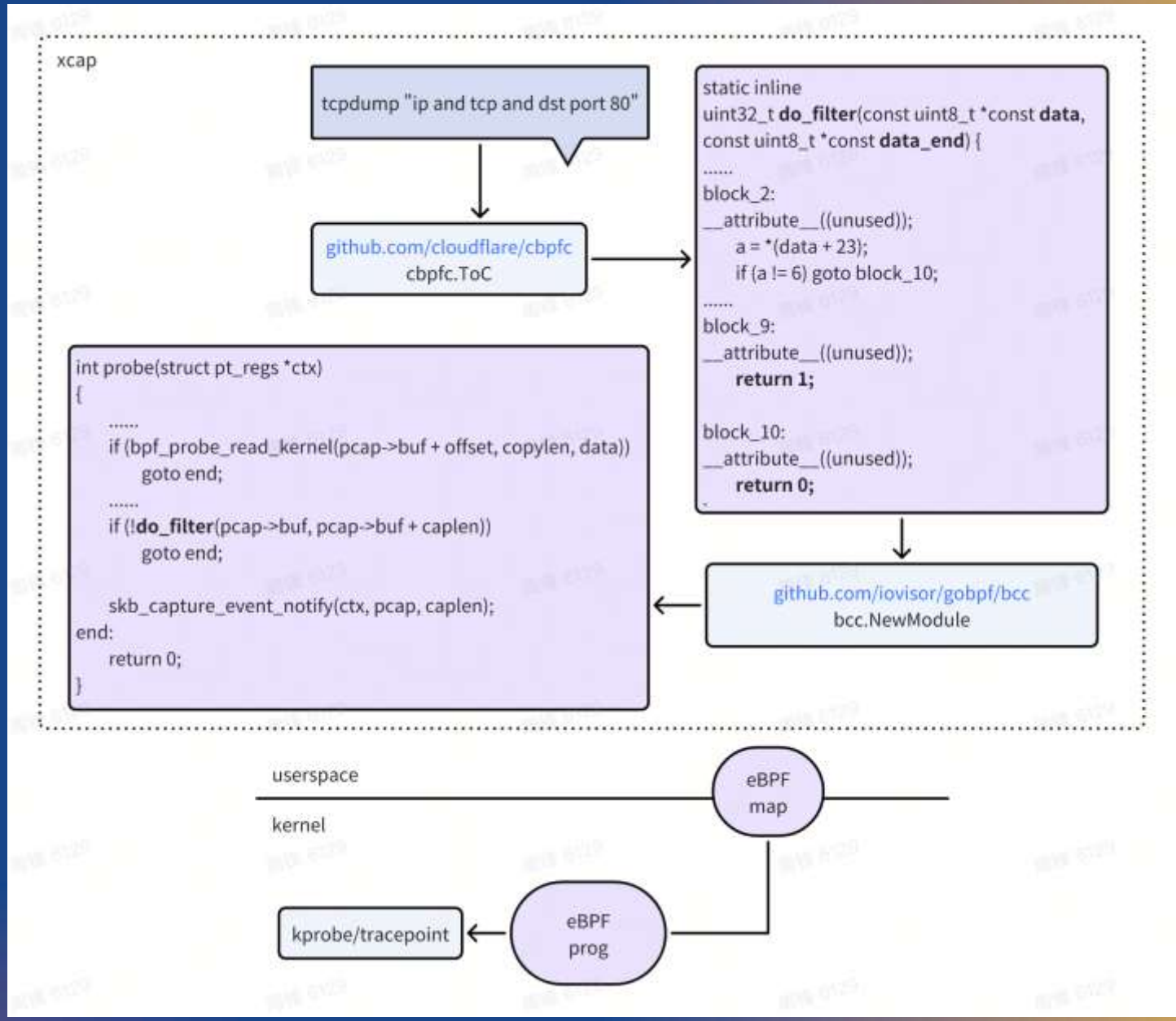
实现原理

bcc集成了llvm的功能，可即时编译生成字节码，并load和attach



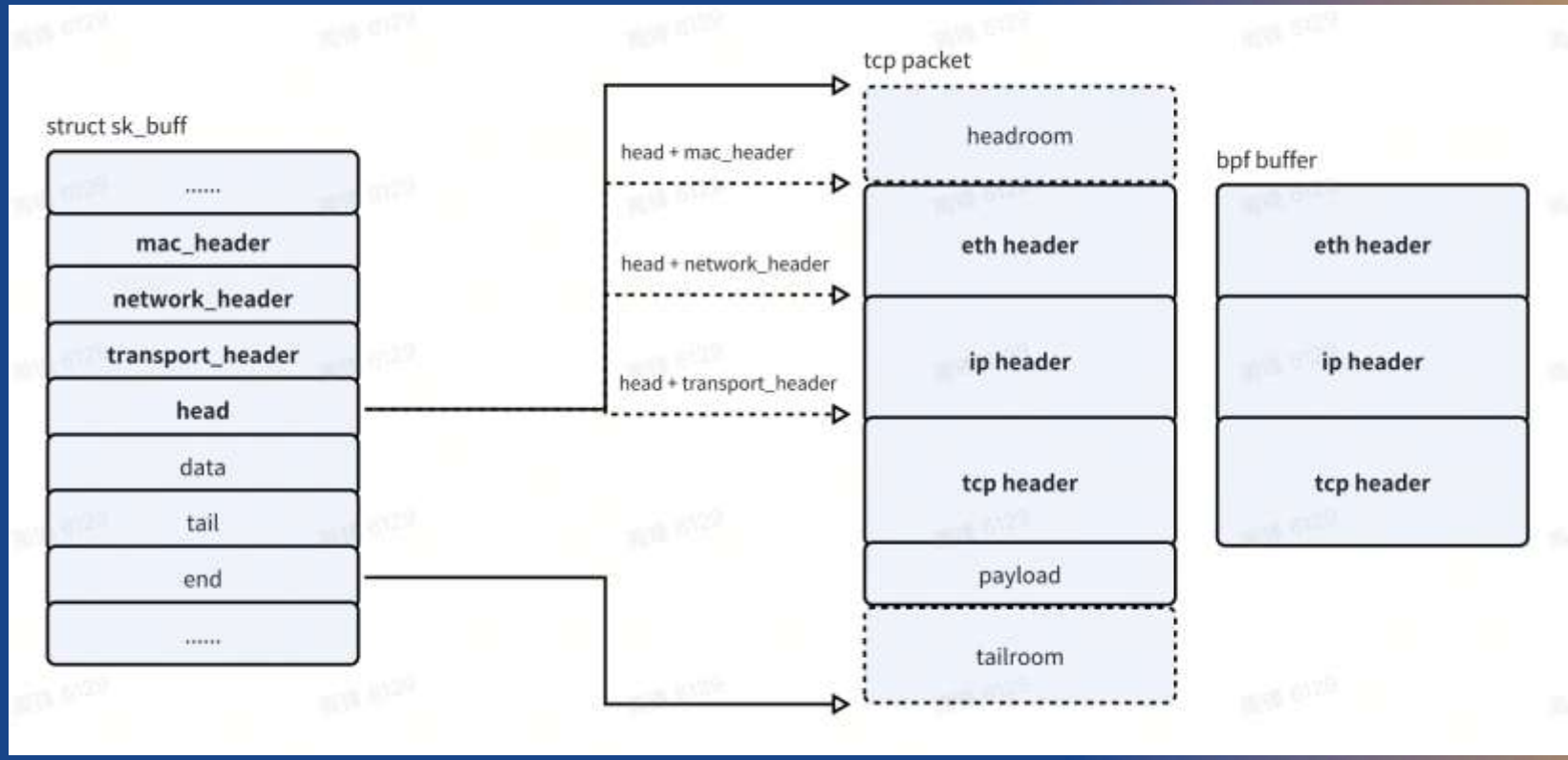
实现原理

tcpdump语法
转换为c函数



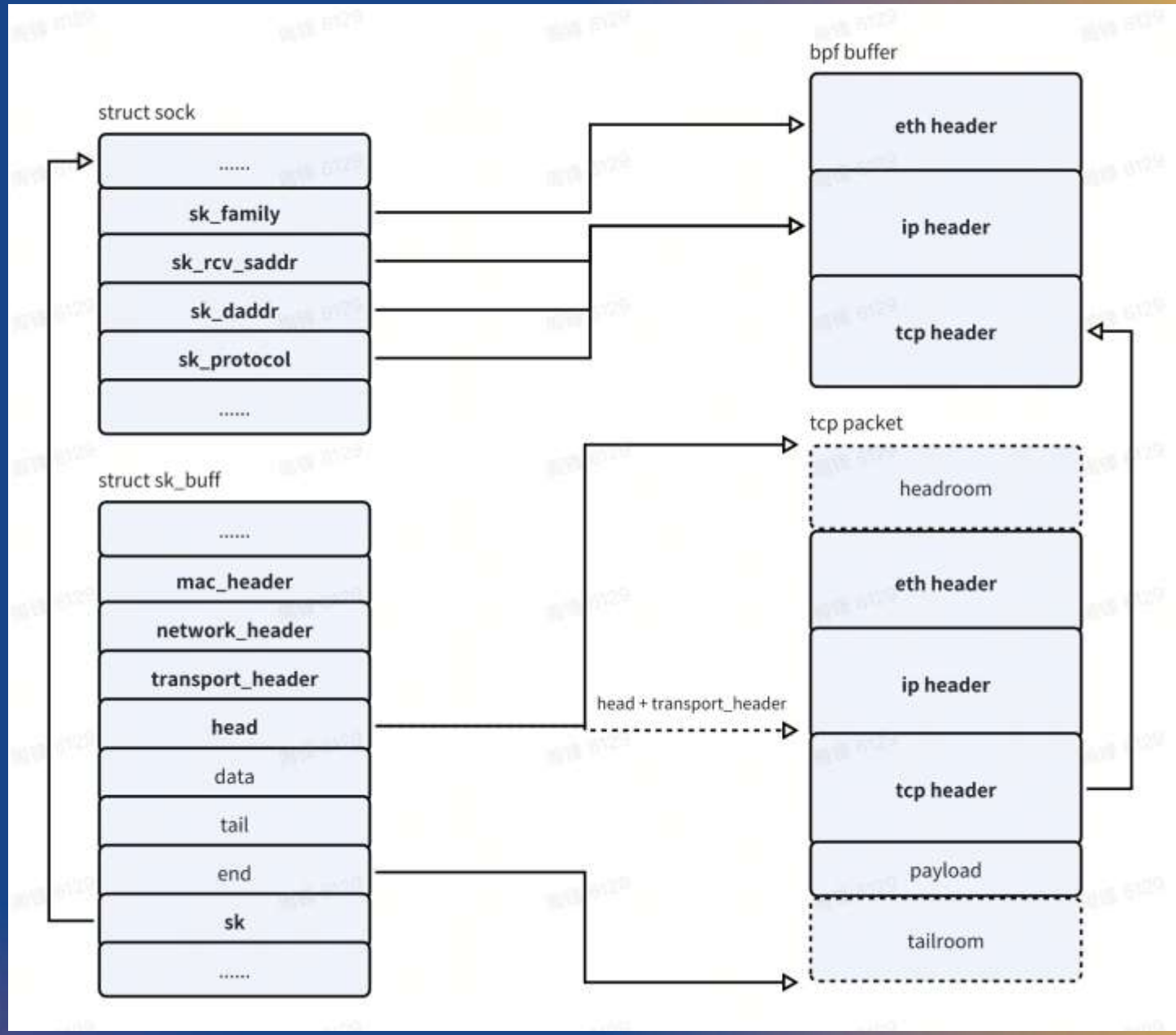
实现原理

skb结构体和报文的对应关系



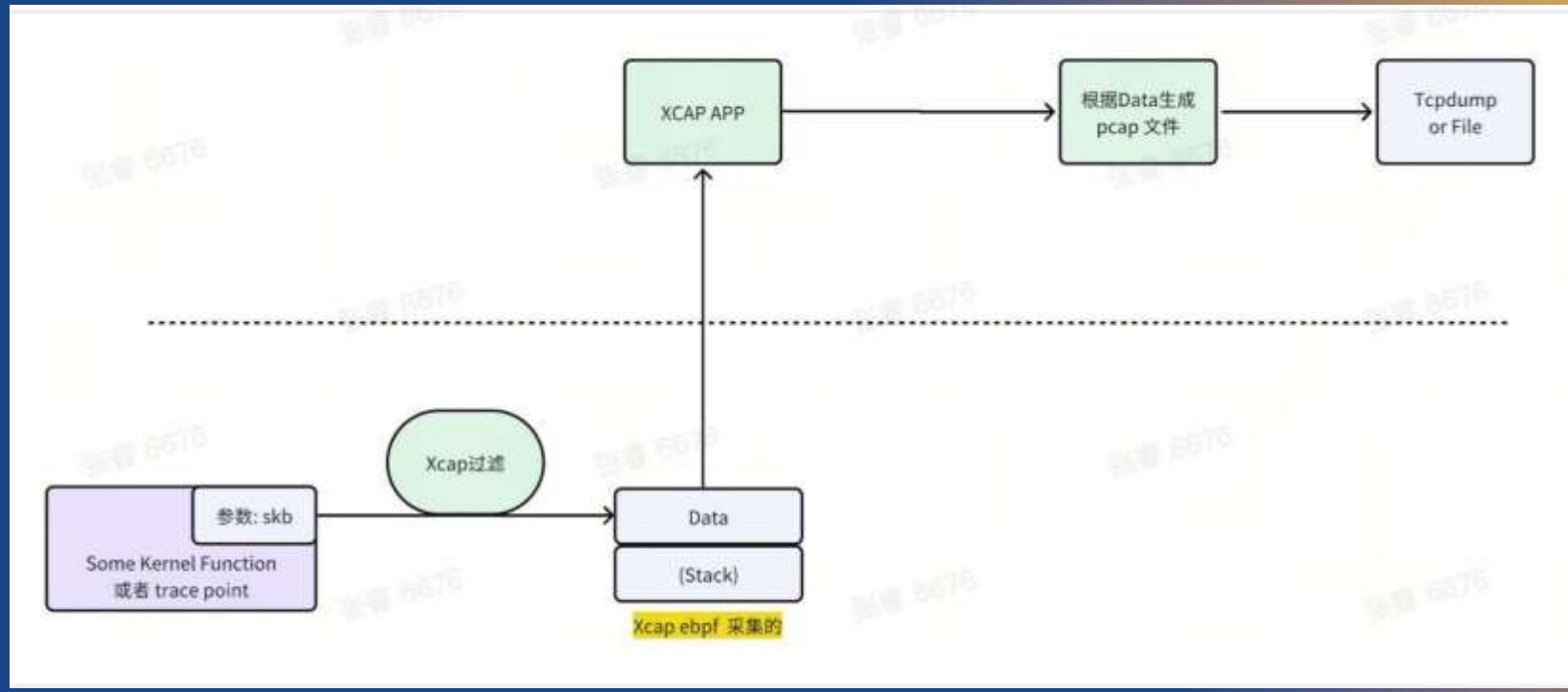
实现原理

用sock伪造skb报文



实现原理

如何将过滤后的报文生成pcap



未来计划

开源计划：

- github开源
- 集成到字节veLinux系统

未来计划

优化方向：

- 使用bpf ringbuf代替perf ringbuf，性能更好，内存消耗更少
- 使用vmlinux btf来自动解析内核函数的参数
- 使用fentry,fexit来降低hook的性能损耗
- 尝试用低开销的uprobe技术，支持DPDK抓包（如：bpftime项目）

Q&A

欢迎关注【字节跳动SYS Tech】公众号

我们是**字节跳动 STE 团队** (System Technologies&Engineering，系统技术与工程）， 聚焦系统技术领域的前沿技术动态，技术创新与实践、行业技术热点等，期待与你的交流。