Packet, where are you?

Track in the stack with pwru

Quentin Monnet
<quentin@isovalent.com>

FOSDEM – 2024-02-04

pwru

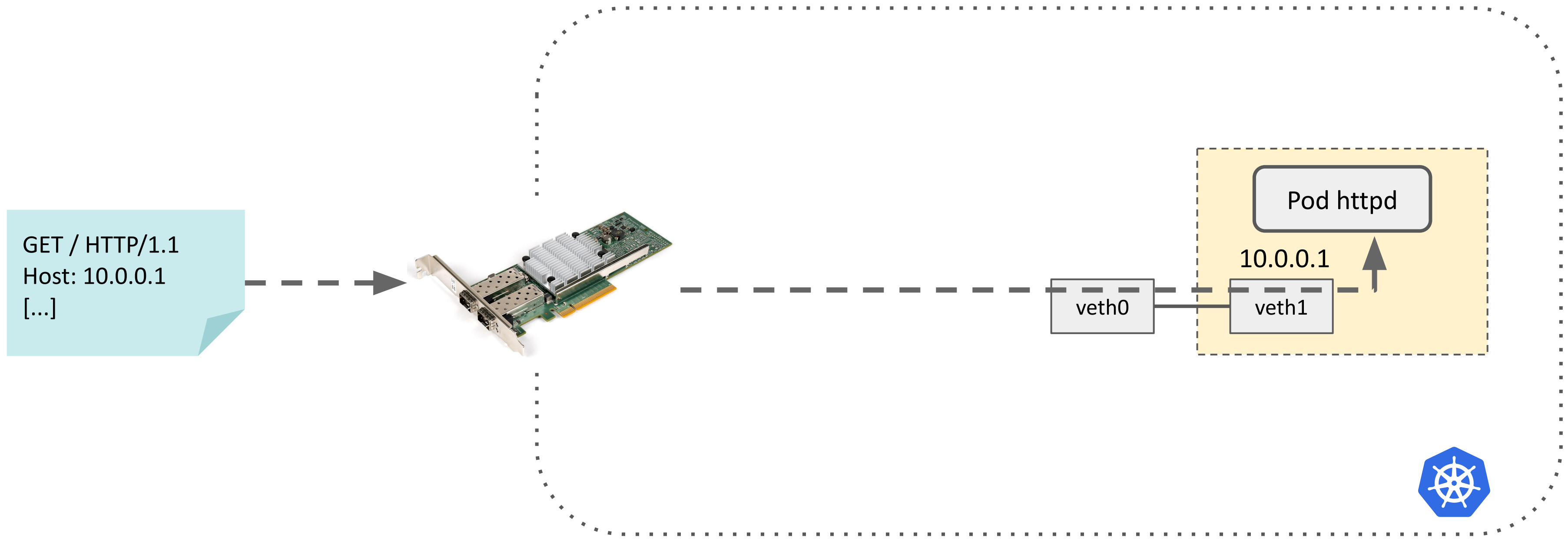
eBPF-based tool to debug packet trajectories in the Linux kernel networking stack

Agenda:

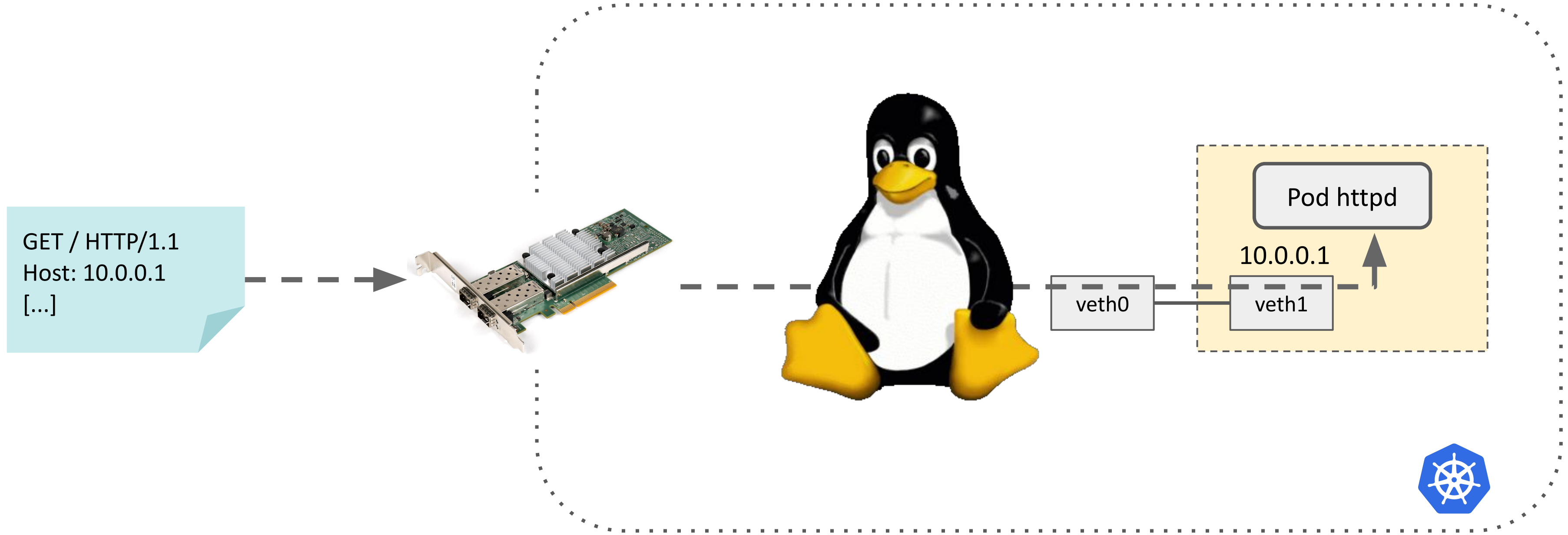
- Problem statement
- Introduction to pwru
- pwru features
- pwru in real life

Disclaimer: I’m a user,
not a project contributor

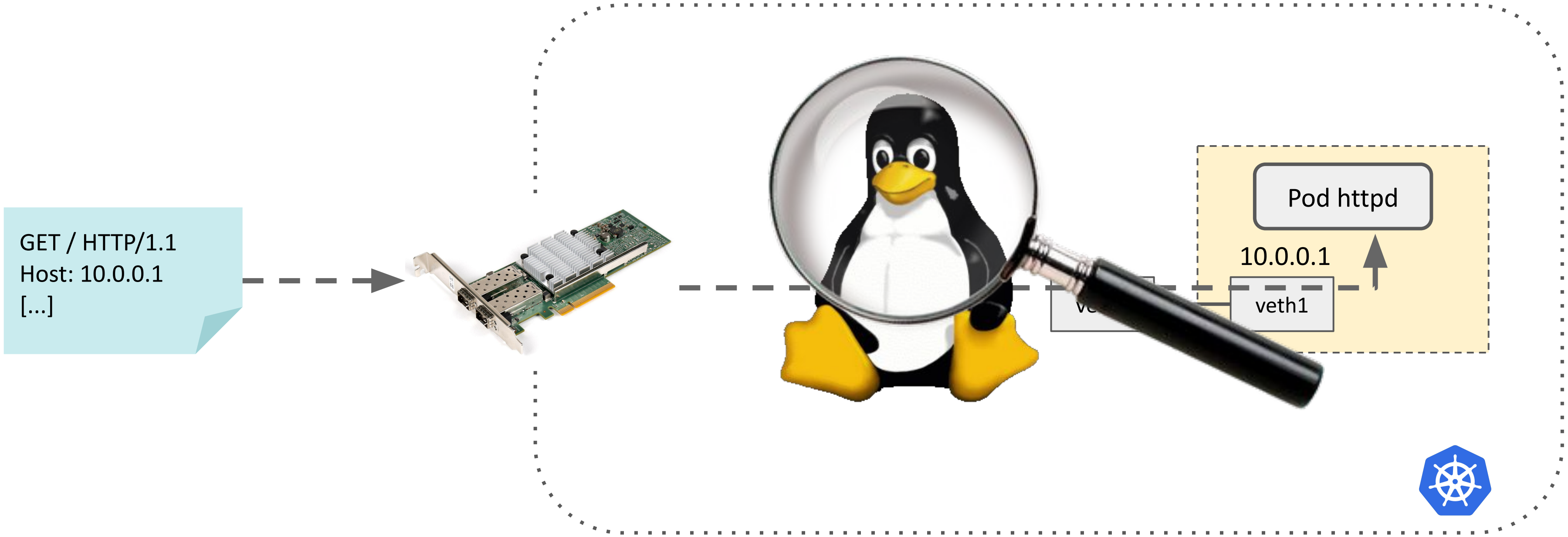
Problem statement



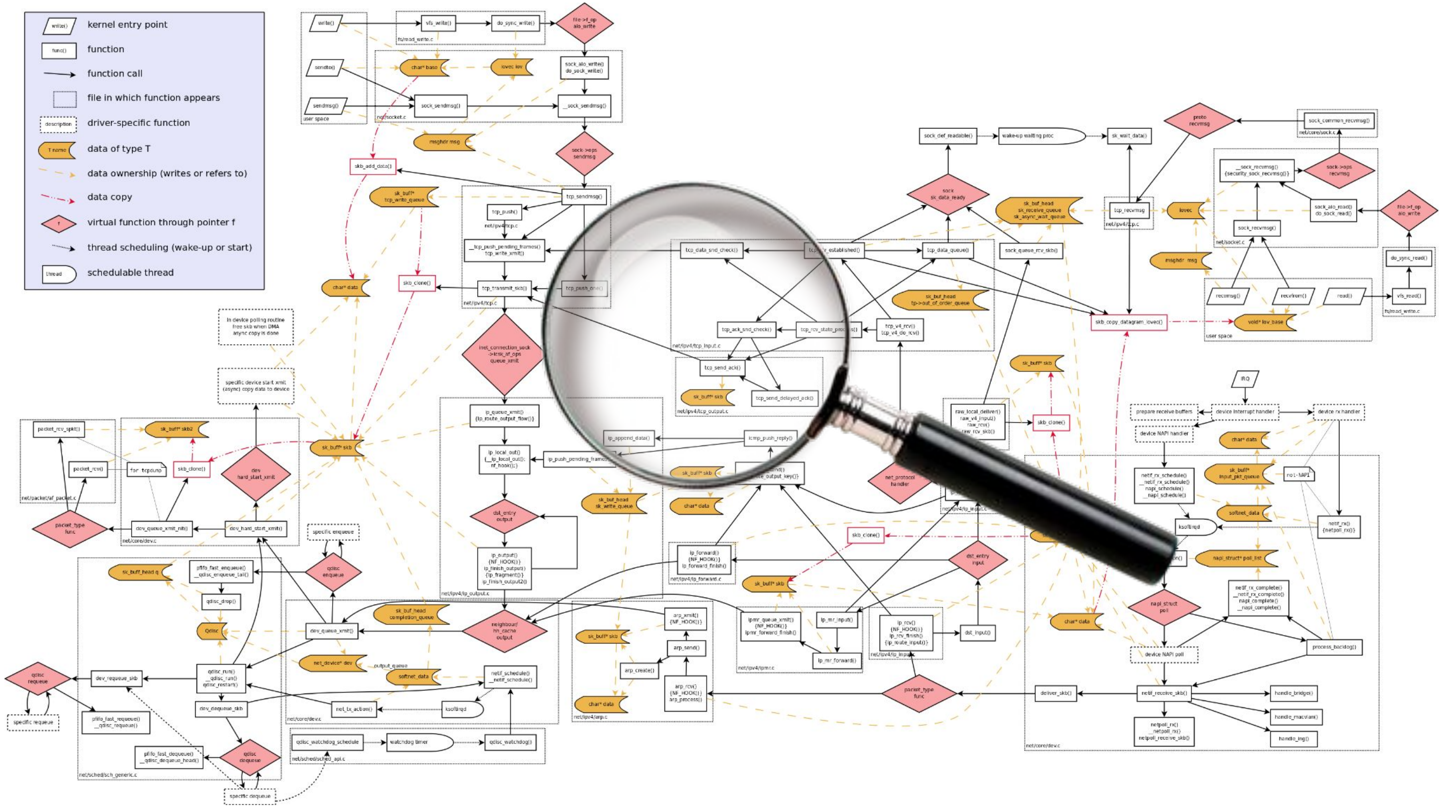
Problem statement



Problem statement

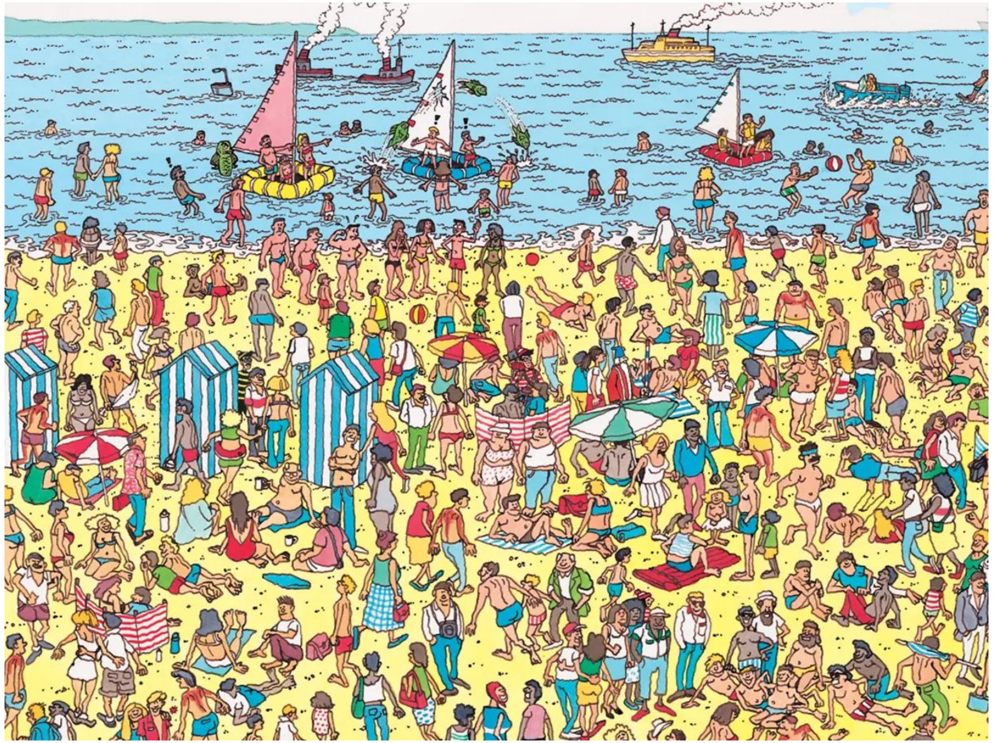


Problem statement


kernel entry point
function
function call
file in which function appears
driver-specific function
data of type T
data ownership (writes or refers to)
data copy
virtual function through pointer f
thread scheduling (wake-up or start)
schedulable thread

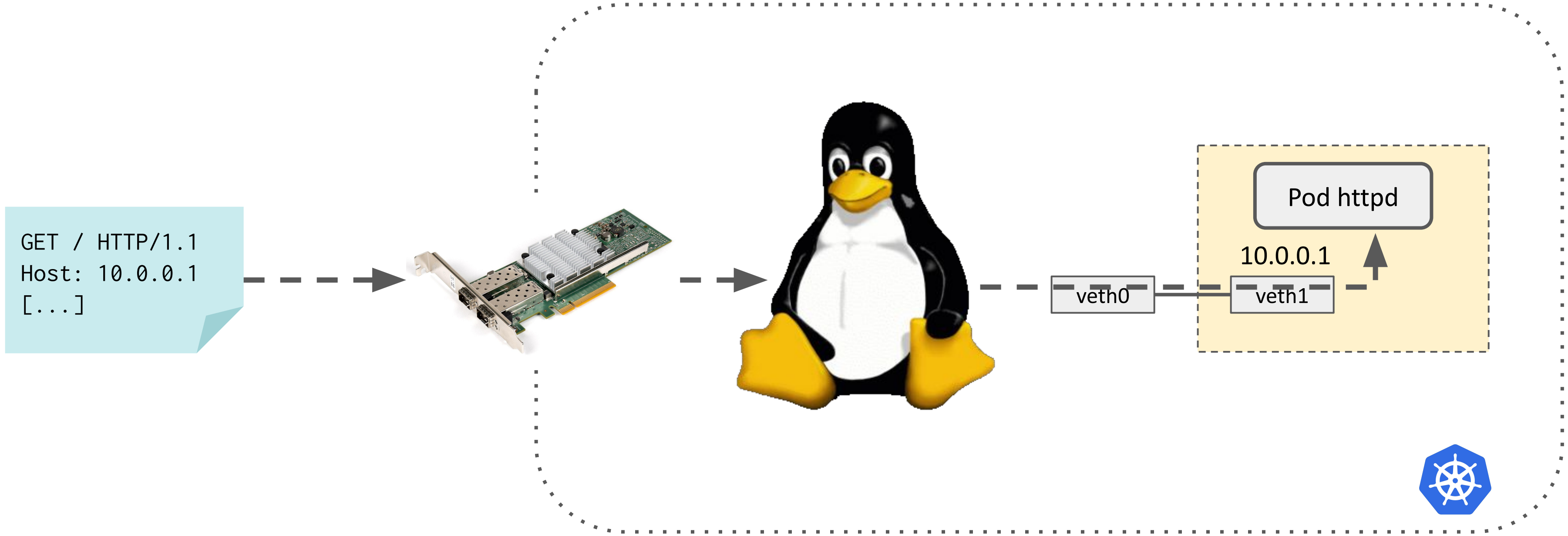
http://web.archive.org/web/20171027184742/https://wiki.linuxfoundation.org/images/1/1c/Network_data_flow_through_kernel.png

Problem statement



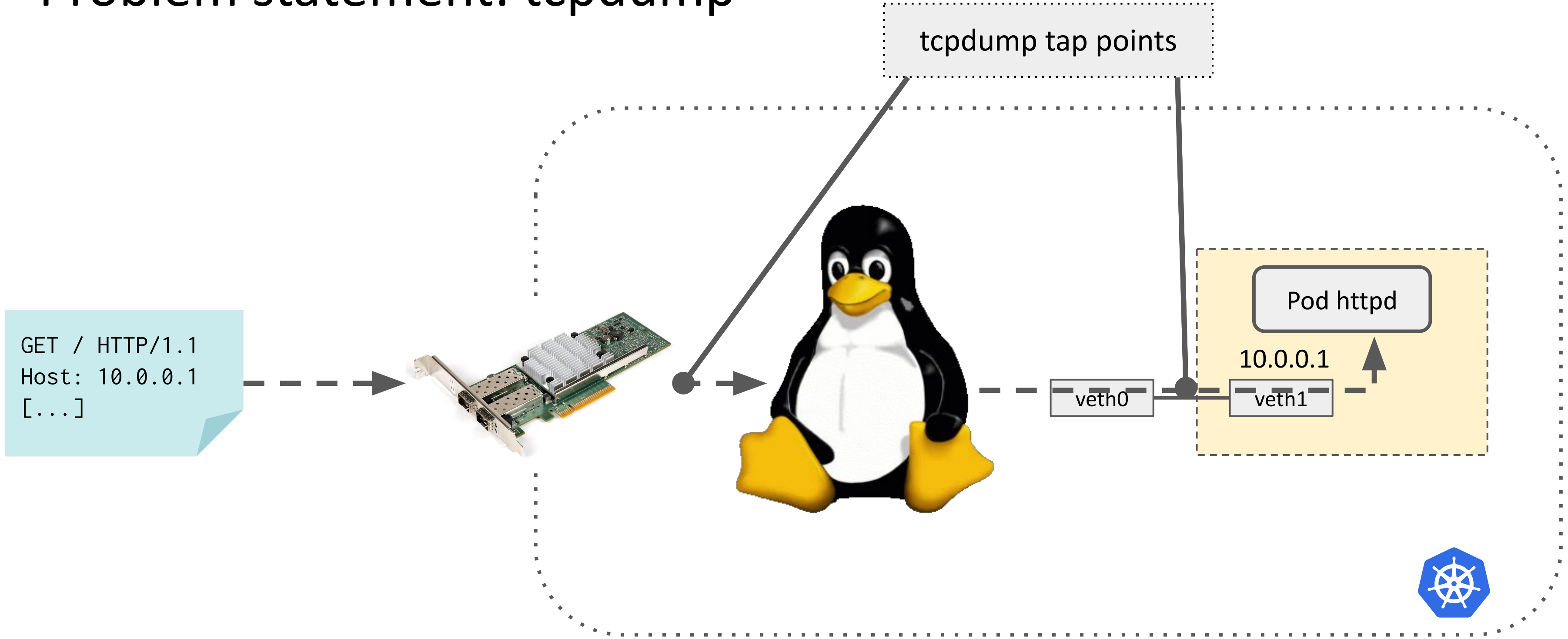
Walker Books

Problem statement: tcpdump



```
# tcpdump -i any 'host 10.0.0.1' ?
```

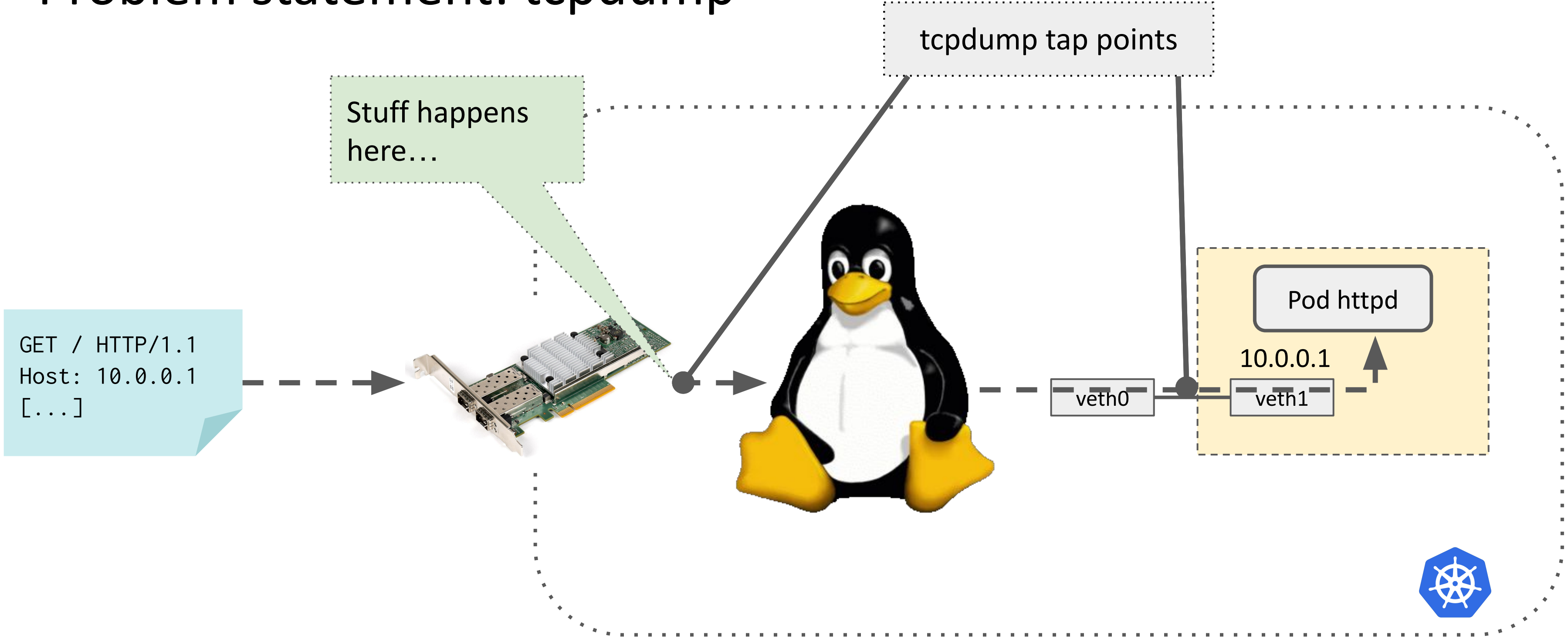
Problem statement: tcpdump



```
GET / HTTP/1.1
Host: 10.0.0.1
[...]
```

```
# tcpdump -i any 'host 10.0.0.1' ?
```

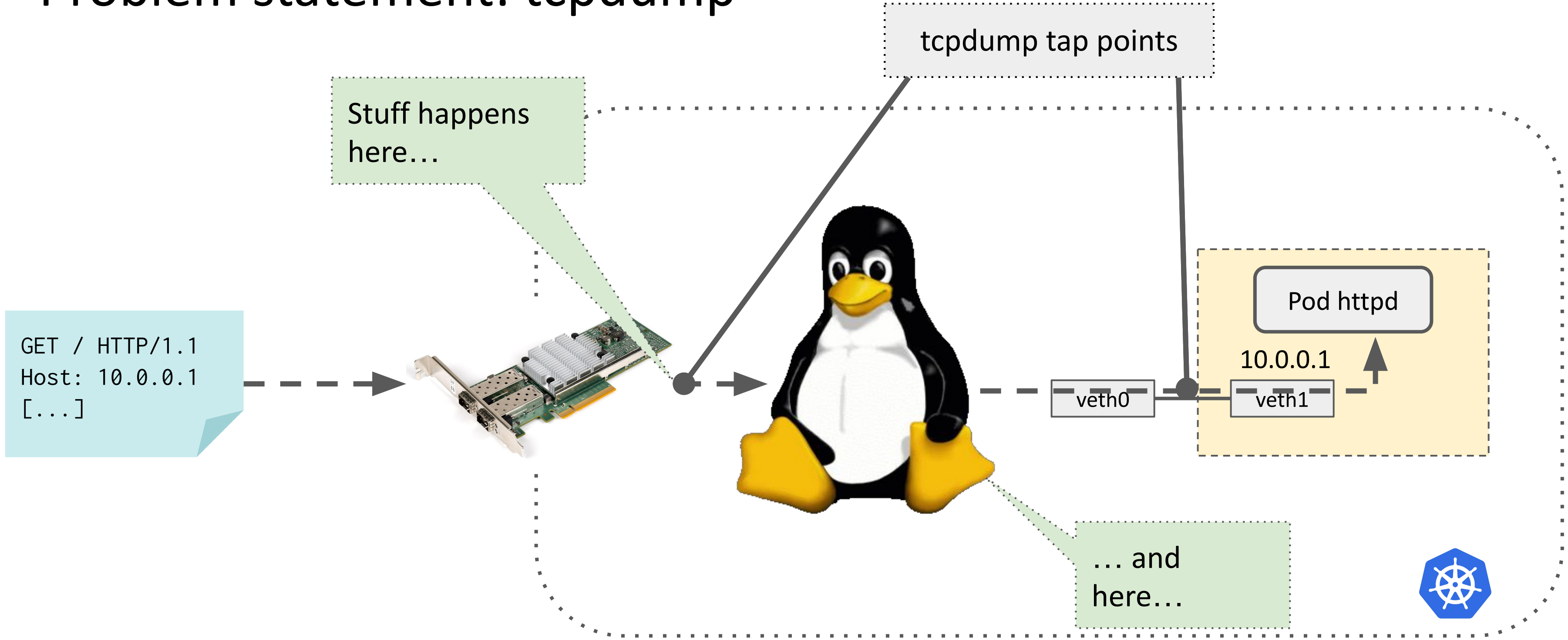
Problem statement: tcpdump



```
GET / HTTP/1.1
Host: 10.0.0.1
[...]
```

```
# tcpdump -i any 'host 10.0.0.1' ?
```

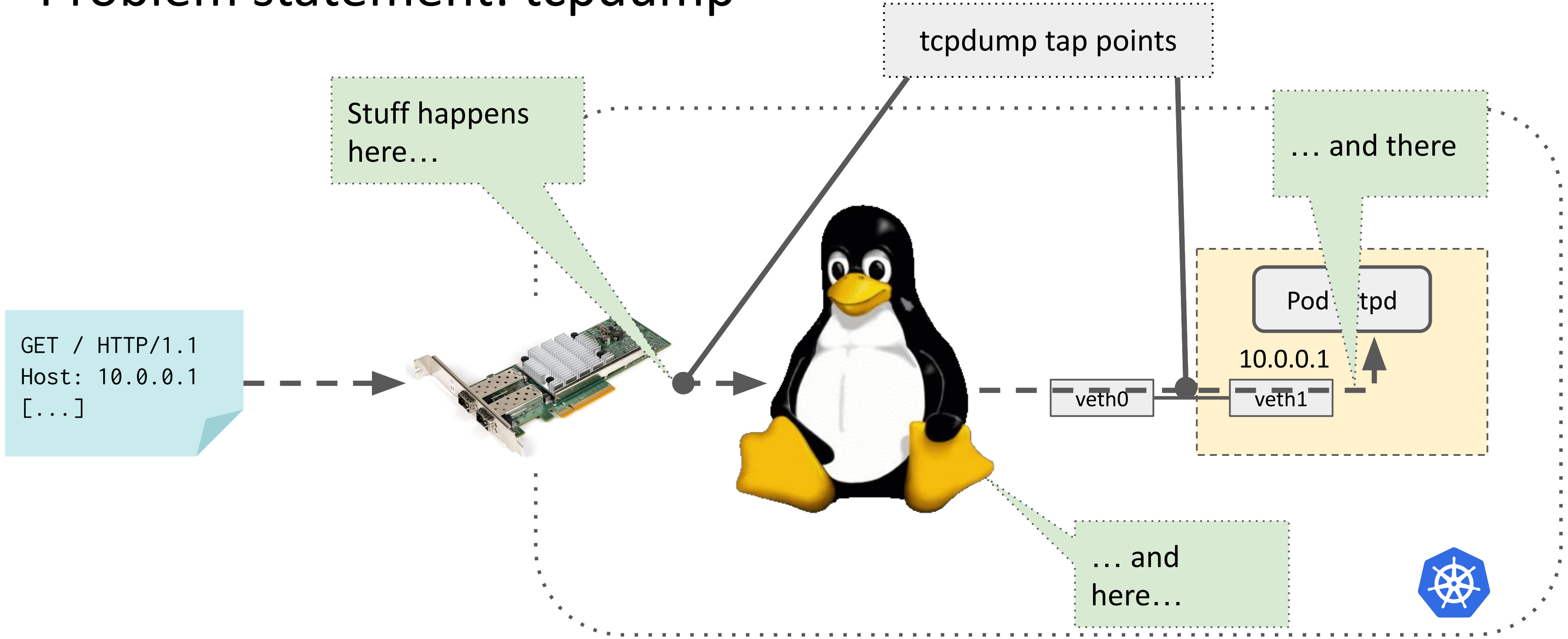
Problem statement: tcpdump



```
GET / HTTP/1.1
Host: 10.0.0.1
[...]
```

```
# tcpdump -i any 'host 10.0.0.1' ?
```

Problem statement: tcpdump



```
# tcpdump -i any 'host 10.0.0.1' ?
```

Problem statement: printk

printk() ?

Problem statement: printk

printk() ?

- × Requires recompiling the kernel
- × Needs reboot in many cases
- × Might panic
- × Many iterations (= very slow debugging)
- × How to filter only particular traffic?

Problem statement: perf (or similar)

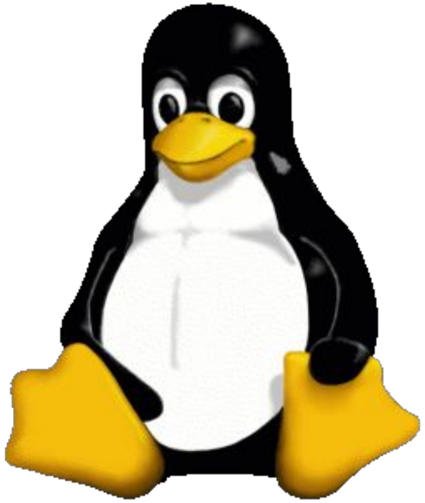
```
# perf record -g -a -e
skb:$KERNEL_FUNC
# perf script
etcd  1234:  skb:kfree_skb: skbaddr=0x...
protocol=2048
    0x8e46b199 kfree_skb+0x79
    0x8e46b199 kfree_skb+0x79
    0x8e473343 sk_stream_kill_queues+0x53
    0x8e535d55 inet_csk_destroy_sock+0x55
    0x8e548cb7 tcp_fin+0x117
    0x8e549829 tcp_data_queue+0x8c9
    [...]
```

Problem statement: perf (or similar)

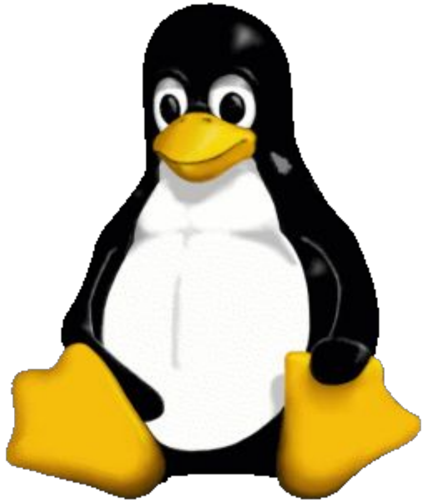
```
# perf record -g -a -e
skb:$KERNEL_FUNC
# perf script
etcd  1234:  skb:kfree_skb: skbaddr=0x...
protocol=2048
    0x8e46b199 kfree_skb+0x79
    0x8e46b199 kfree_skb+0x79
    0x8e473343 sk_stream_kill_queues+0x53
    0x8e535d55 inet_csk_destroy_sock+0x55
    0x8e548cb7 tcp_fin+0x117
    0x8e549829 tcp_data_queue+0x8c9
    [...]
```

- × Very limited filtering (e.g., cannot specify udp.port=53)
- × Lot of noise
- × Which functions to trace?

What if...

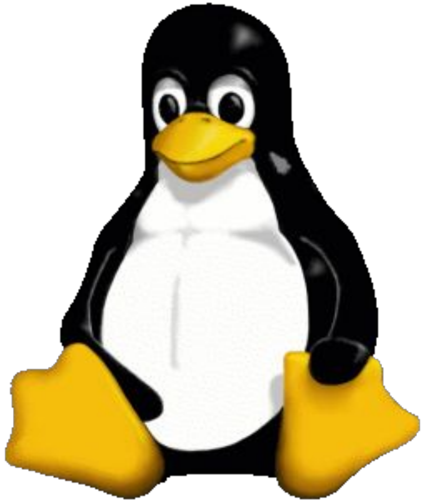


What if...



Get a list of all packet processing functions?

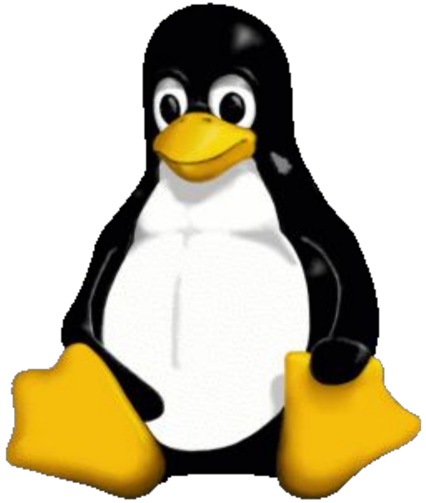
What if...



Get a list of all packet processing functions?

Get callbacks when these functions are executed?

What if...



Get a list of all packet processing functions?

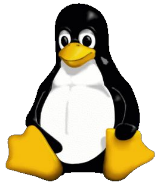
Get callbacks when these functions are executed?

Filter callbacks only for traffic of interest?

Packet Where R U? - Leveraging eBPF

Programmable, **performant**, and **safe**
in-kernel execution environment that runs
native code on certain events or hooks

```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```



user space

kernel

Packet in the kernel

```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

user space

kernel

Packet in the kernel

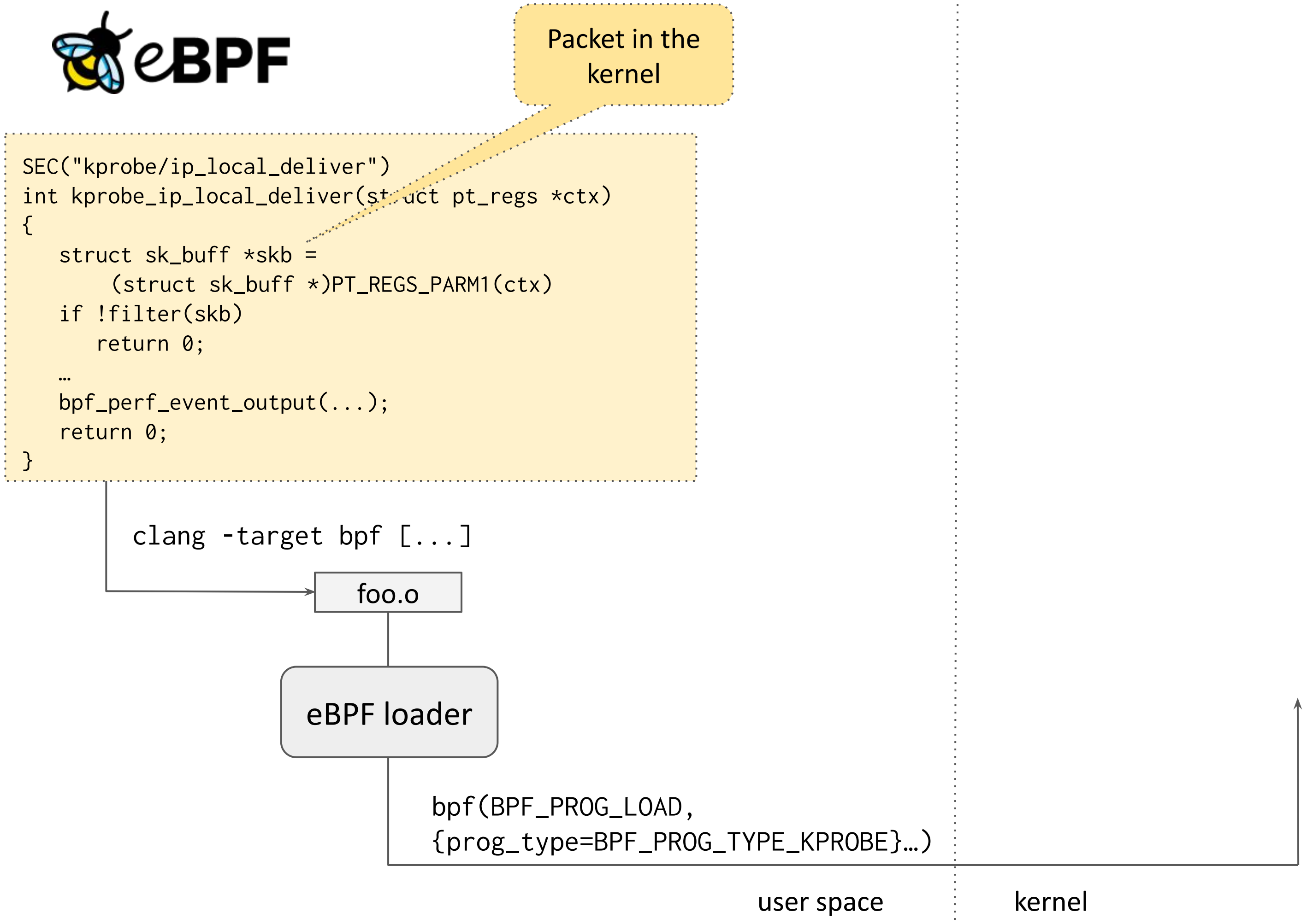
```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

foo.o

user space

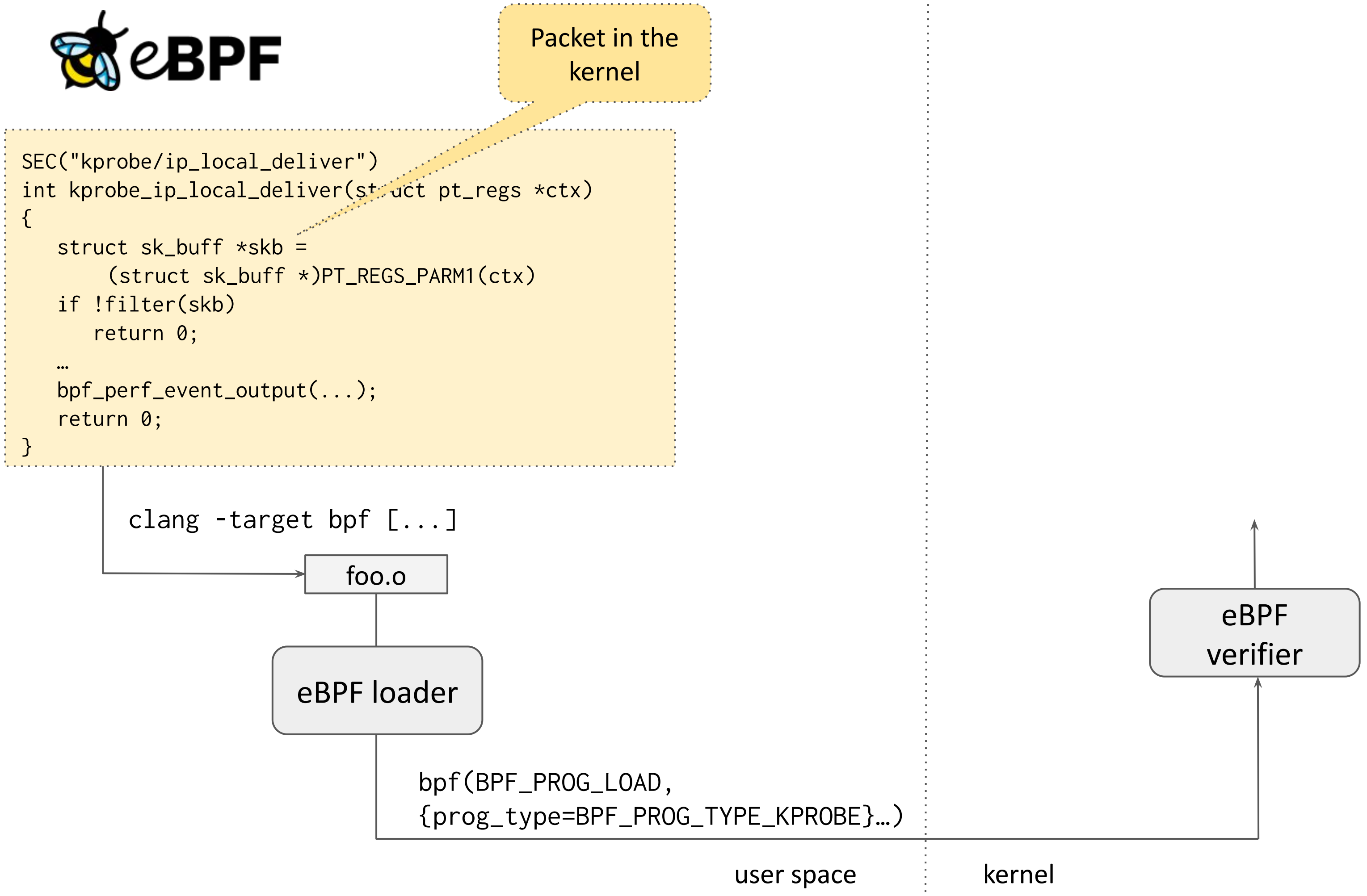
kernel



```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

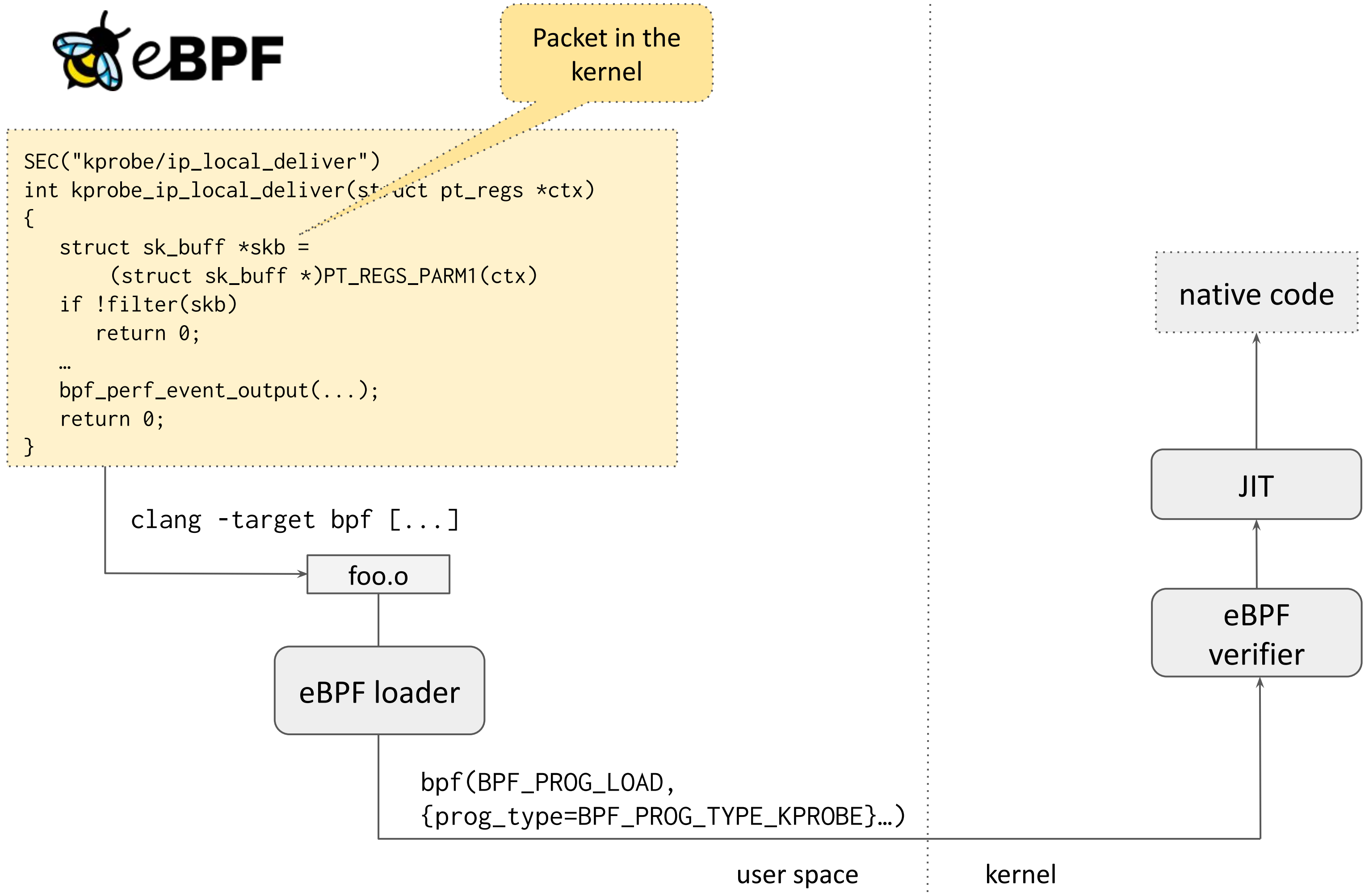
```
bpf(BPF_PROG_LOAD,
{prog_type=BPF_PROG_TYPE_KPROBE}…)
```



```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

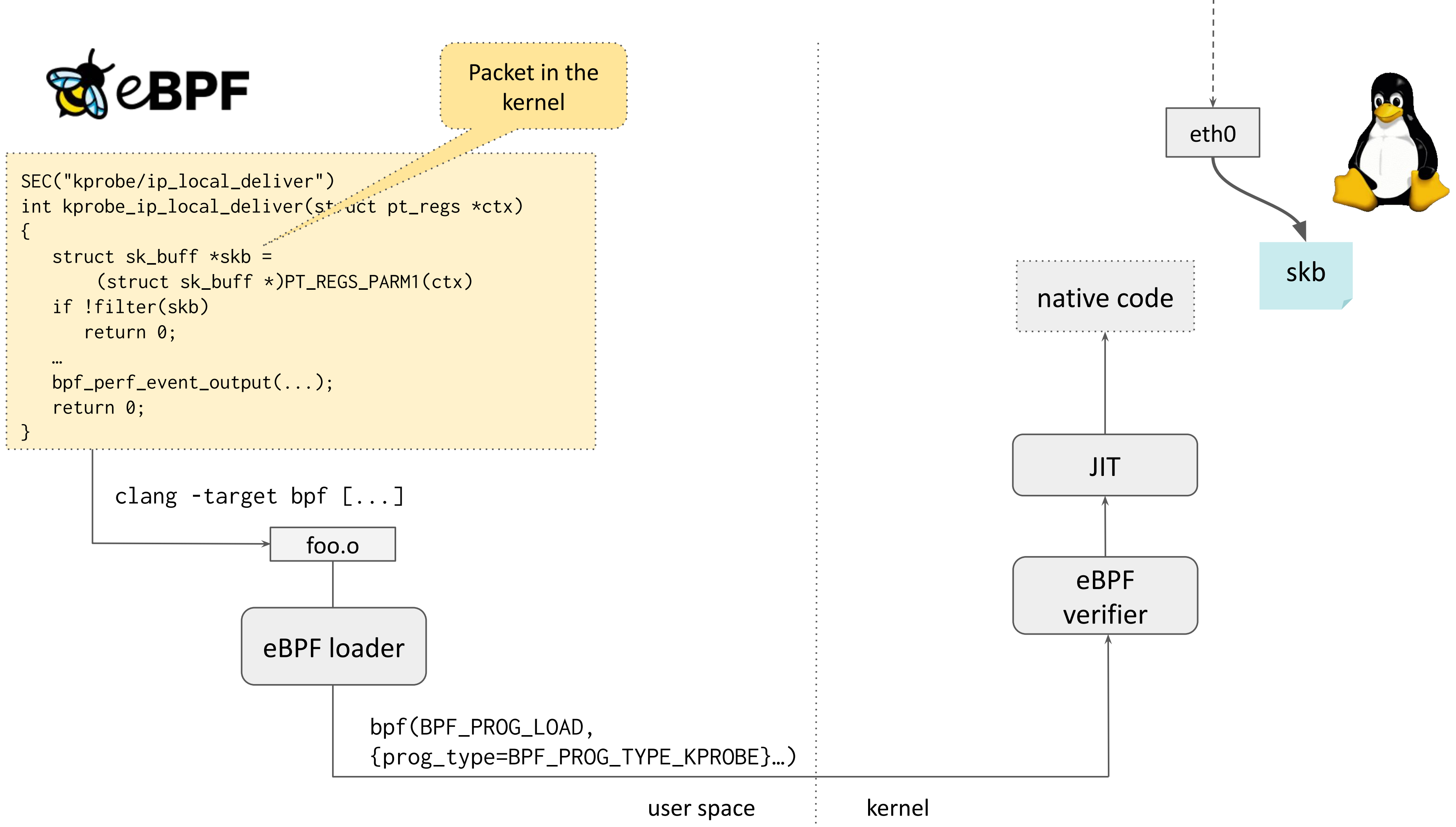
```
bpf(BPF_PROG_LOAD,
{prog_type=BPF_PROG_TYPE_KPROBE}…)
```



```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

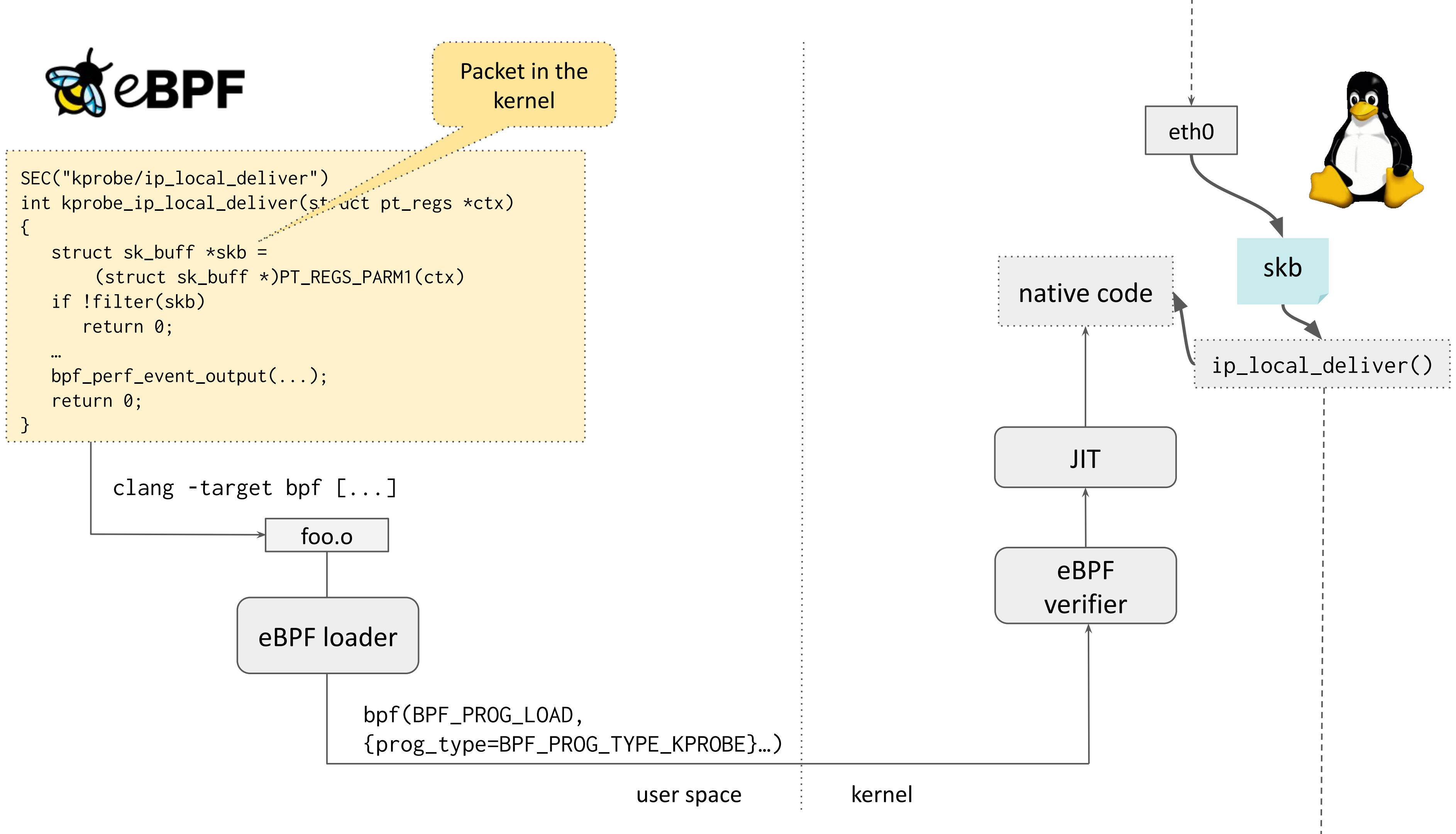
```
bpf(BPF_PROG_LOAD,
{prog_type=BPF_PROG_TYPE_KPROBE}…)
```



```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

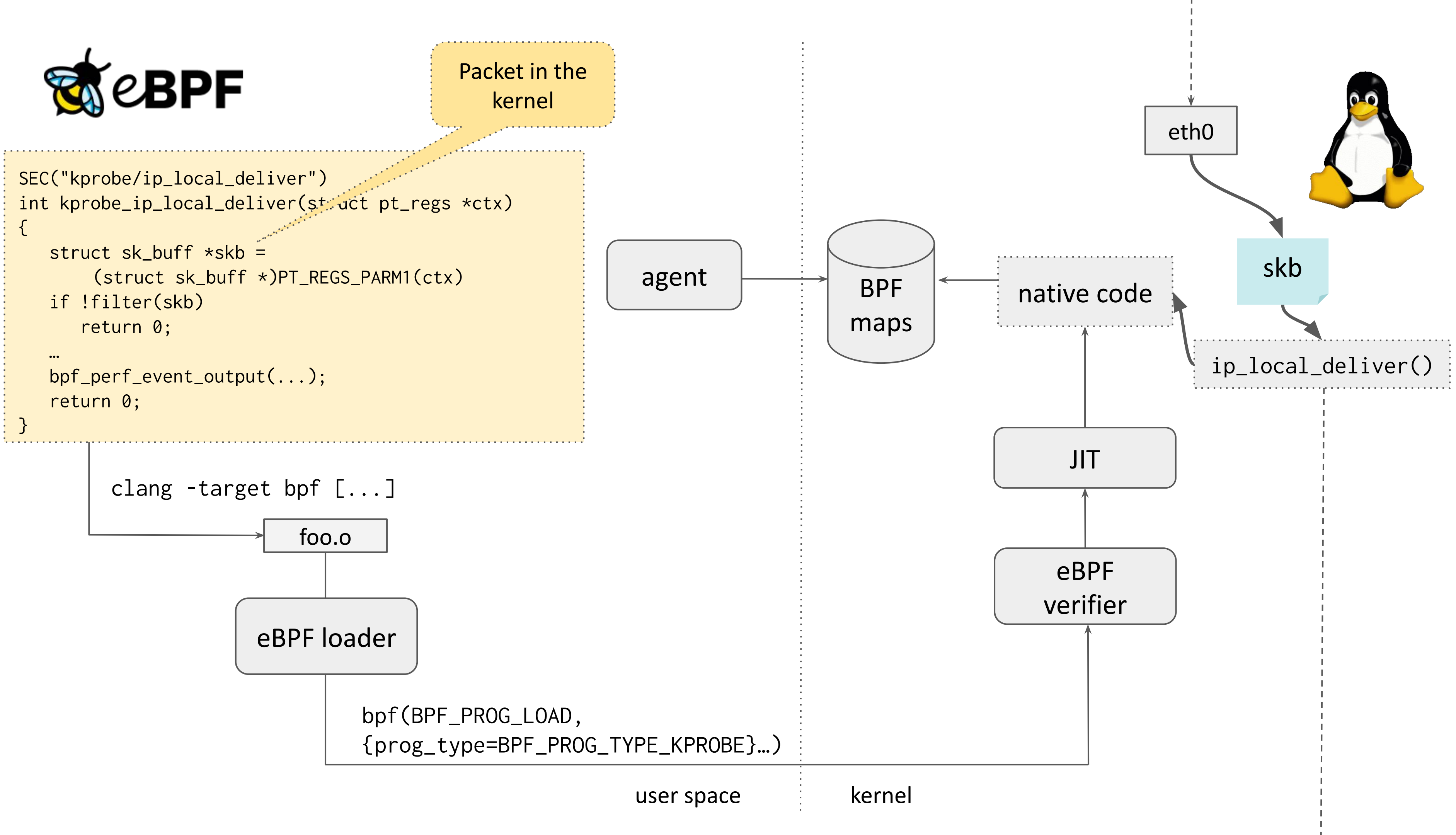
```
bpf(BPF_PROG_LOAD,
{prog_type=BPF_PROG_TYPE_KPROBE}…)
```



```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

```
bpf(BPF_PROG_LOAD,
{prog_type=BPF_PROG_TYPE_KPROBE}…)
```



```
SEC("kprobe/ip_local_deliver")
int kprobe_ip_local_deliver(struct pt_regs *ctx)
{
   struct sk_buff *skb =
       (struct sk_buff *)PT_REGS_PARM1(ctx)
   if !filter(skb)
      return 0;
   …
   bpf_perf_event_output(...);
   return 0;
}
```

```
clang -target bpf [...]
```

```
bpf(BPF_PROG_LOAD,
{prog_type=BPF_PROG_TYPE_KPROBE}…)
```

How do we keep track of packet processing functions in the Linux kernel?

BPF Type Format (BTF)

*“We developed an algorithm that compresses **124MB** of DWARF type data into just **1.5MB** of compact BTF type data, making it suitable to be included in the Linux kernel image by default.”*
https://facebookmicrosites.github.io/bpf/blog/2018/11/14/btf-enhancement.html

- Metadata format with debug information like function signature
- Kernel BTF available via /sys/kernel/btf/vmlinux

BPF Type Format (BTF)

*“We developed an algorithm that compresses **124MB** of DWARF type data into just **1.5MB** of compact BTF type data, making it suitable to be included in the Linux kernel image by default.”*
https://facebookmicrosites.github.io/bpf/blog/2018/11/14/btf-enhancement.html

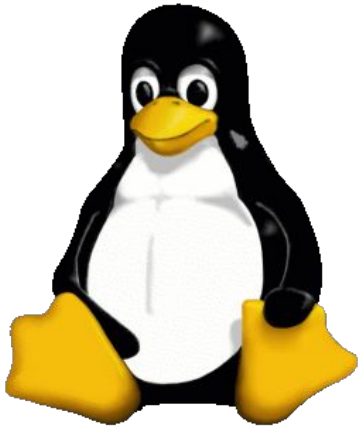
- Metadata format with debug information like function signature
- Kernel BTF available via /sys/kernel/btf/vmlinux

```
u32
skb_get_mark(struct __sk_buff *skb) {
    return skb->mark;
}
```

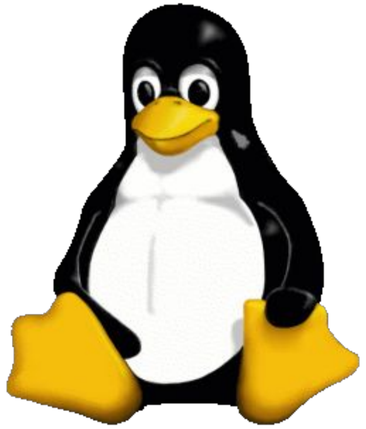
```
$ bpftool btf dump file foo.o
```

```
[1] PTR '(anon)' type_id=2
[2] STRUCT '__sk_buff' size=184 vlen=32
      'len' type_id=3 bits_offset=0
      'pkt_type' type_id=3 bits_offset=32
      'mark' type_id=3 bits_offset=64
...
[23] FUNC_PROTO '(anon)' ret_type_id=15 vlen=1
      'skb' type_id=1
[24] FUNC 'skb_get_mark' type_id=23 linkage=global
```

pwru



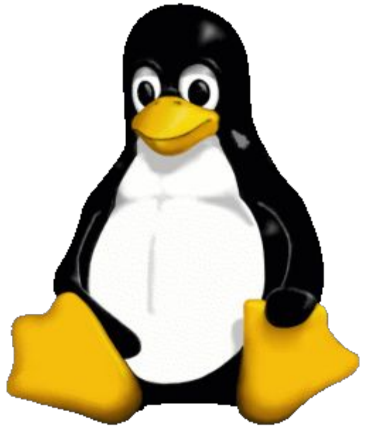
pwru



Get all functions which accept SKB from BTF file

(Get a list of all packet processing functions?)

pwru



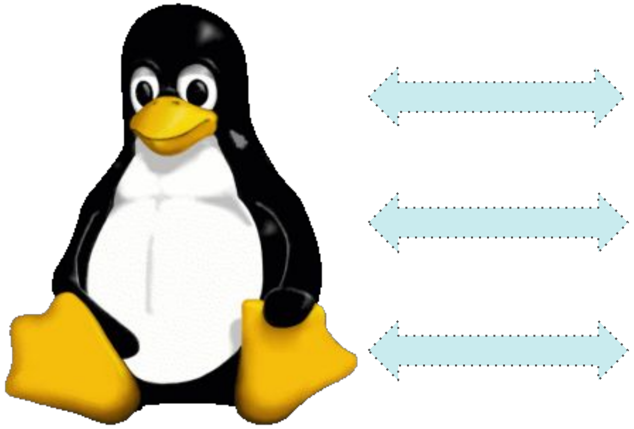
Get all functions which accept SKB from BTF file

(Get a list of all packet processing functions?)

Attach k(ret)probes to all of them

(Get callbacks when these functions are executed?)

pwru



Get all functions which accept SKB from BTF file

(Get a list of all packet processing functions?)

Attach k(ret)probes to all of them

(Get callbacks when these functions are executed?)

Filter packets with eBPF

(Filter callbacks only for traffic of interest?)

What does it look like?

#

#

```
# iptables -t filter -I OUTPUT 1 -m tcp --proto tcp --dst 1.1.1.1/32 -j DROP
#
```

```
#
```

```
# iptables -t filter -I OUTPUT 1 -m tcp --proto tcp --dst 1.1.1.1/32 -j DROP
#
```

```
# pwru 'dst host 1.1.1.1 and tcp and dst port 80'
# iptables -t filter -I OUTPUT l -m tcp --proto tcp --dst 1.1.1.1/32 -j DROP
#
```

```
# pwru 'dst host 1.1.1.1 and tcp and dst port 80'
2024/02/04 02:19:22 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [---------------------------------------------------------------------------] 100.00% 52449 p/s
2024/02/04 02:19:22 Attached (ignored 0)
2024/02/04 02:19:22 Listening for events..
               SKB  CPU          PROCESS                 FUNC
```

```
# iptables -t filter -I OUTPUT 1 -m tcp --proto tcp --dst 1.1.1.1/32 -j DROP
# curl 1.1.1.1
```

```
# pwru 'dst host 1.1.1.1 and tcp and dst port 80'
2024/02/04 02:19:22 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [----------------------------------------------------------------------------] 100.00% 52449 p/s
2024/02/04 02:19:22 Attached (ignored 0)
2024/02/04 02:19:22 Listening for events..
               SKB  CPU          PROCESS                 FUNC
```

```
# iptables -t filter -I OUTPUT 1 -m tcp --proto tcp --dst 1.1.1.1/32 -j DROP
# curl 1.1.1.1
^C
```

```
# pwru 'dst host 1.1.1.1 and tcp and dst port 80'
2024/02/04 02:19:22 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [---------------------------------------------------------------------------] 100.00% 52449 p/s
2024/02/04 02:19:22 Attached (ignored 0)
2024/02/04 02:19:22 Listening for events..
               SKB  CPU          PROCESS                 FUNC
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         ip_local_out
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         __ip_local_out
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         nf_hook_slow
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)] kfree_skb_reason(SKB_DROP_REASON_NETFILTER_DROP)
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]   skb_release_head_state
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]               tcp_wfree
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         skb_release_data
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         kfree_skbmem
```

```
# iptables -t filter -I OUTPUT 1 -m tcp --proto tcp --dst 1.1.1.1/32 -j DROP
# curl 1.1.1.1
^C
```

```
# pwru 'dst host 1.1.1.1 and tcp and dst port 80'
2024/02/04 02:19:22 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [-----------------------------------------------------------------------------] 100.00% 52449 p/s
2024/02/04 02:19:22 Attached (ignored 0)
2024/02/04 02:19:22 Listening for events..
               SKB  CPU          PROCESS                 FUNC
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         ip_local_out
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         __ip_local_out
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         nf_hook_slow
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)] kfree_skb_reason(SKB_DROP_REASON_NETFILTER_DROP)
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]   skb_release_head_state
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]               tcp_wfree
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         skb_release_data
0xffff9e0a374a10e8    5 [/usr/bin/curl(16232)]         kfree_skbmem
^C2024/02/04 02:19:29 Received signal, exiting program..
2024/02/04 02:19:29 Detaching kprobes…
5 / 5 [-------------------------------------------------------------------------------------] 100.00% 6 p/s
```

Tell me more!
Some pwru features

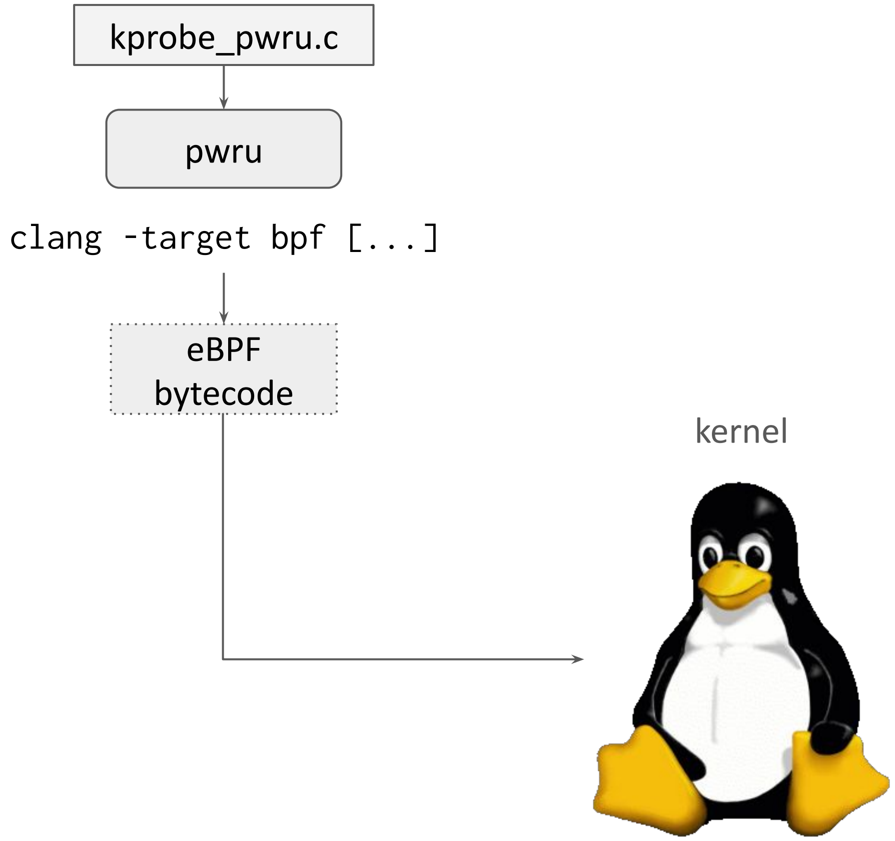
```
$ pwru --help
Usage: pwru [options] [pcap-filter]
   Available pcap-filter: see "man 7 pcap-filter"
   Available options:
     --all-kmods                 attach to all available kernel modules
     --backend string            Tracing backend('kprobe', 'kprobe-multi'). Will auto-detect if not specified.
     --filter-func string        filter kernel functions to be probed by name (exact match, supports RE2
                                 regular expression)
     --filter-ifname string      filter skb ifname in --filter-netns (if not specified, use current netns)
     --filter-mark uint32        filter skb mark
     --filter-netns string       filter netns ("/proc/<pid>/ns/net", "inode:<inode>")
     --filter-trace-tc           trace TC bpf progs
     --filter-track-skb          trace a packet even if it does not match given filters (e.g., after NAT
                                 or tunnel decapsulation)
 -h, --help                      display this message and exit
     --kernel-btf string         specify kernel BTF file
     --kmods strings             list of kernel modules names to attach to
     --output-file string        write traces to file
     --output-limit-lines uint   exit the program after the number of events has been received/printed
     --output-meta               print skb metadata
     --output-skb                print skb
     --output-stack              print stack
     --output-tuple              print L4 tuple
     --timestamp string          print timestamp per skb ("current", "relative", "absolute", "none" (default
                                 "none")
     --version                   show pwru version and exit
```

```
$ pwru --help
Usage: pwru [options] [pcap-filter]
    Available pcap-filter: see "man 7 pcap-filter"
    Available options:
```

Pcap-filter support

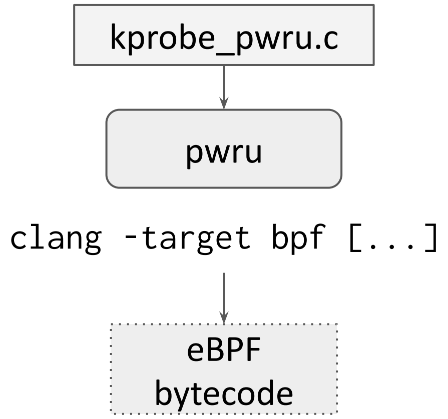
Packet filtering

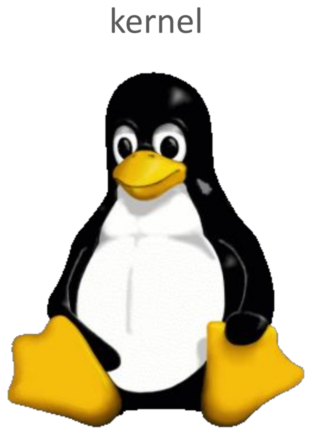
```
# pwru
```



Packet filtering

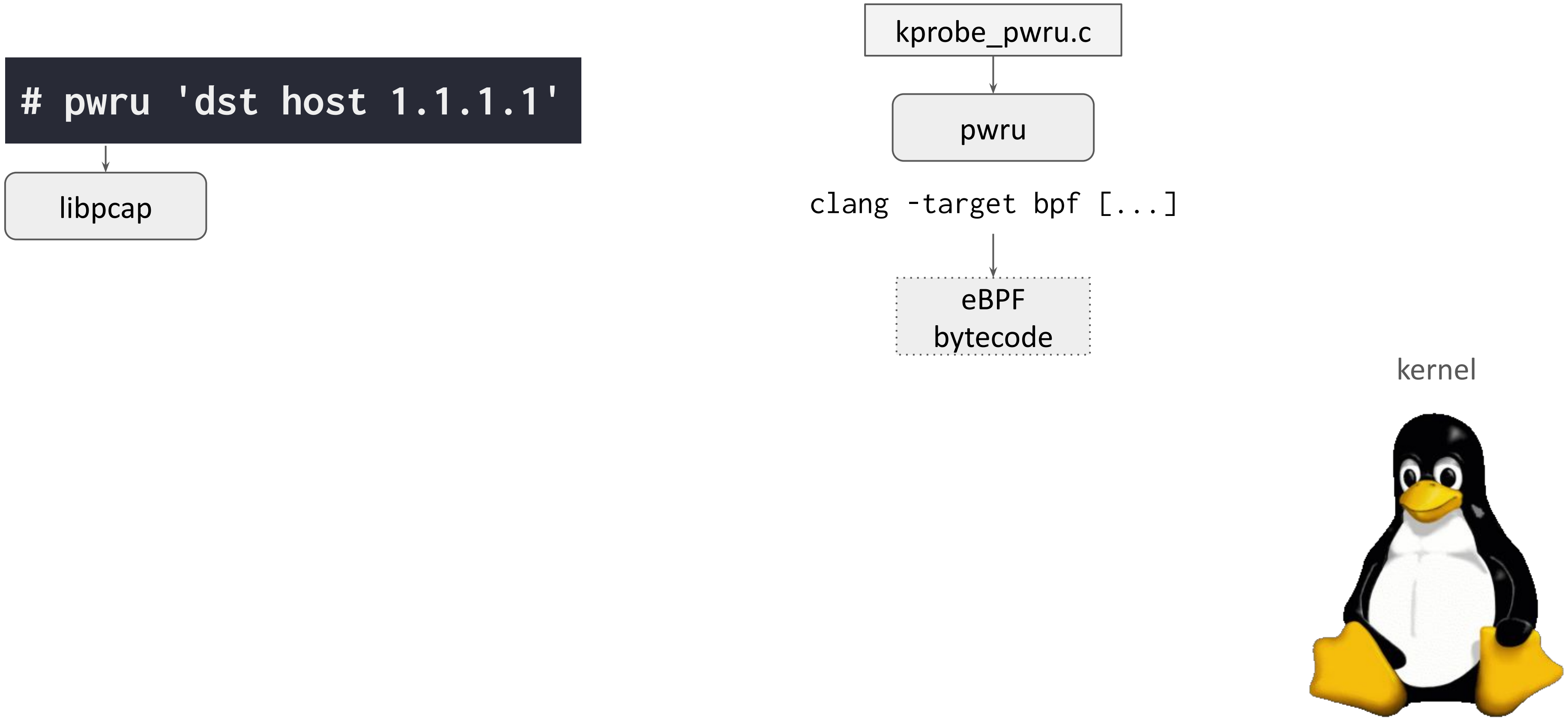
```
# pwru 'dst host 1.1.1.1'
```





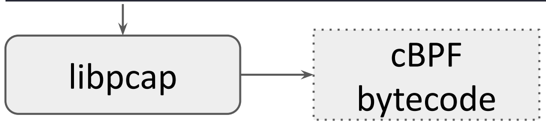
Packet filtering

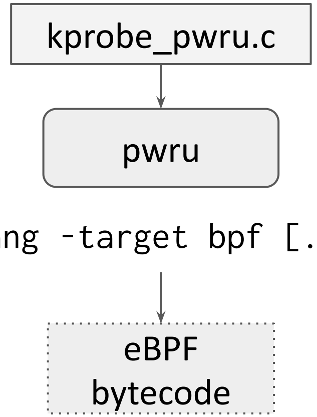
```
# pwru 'dst host 1.1.1.1'
```



Packet filtering

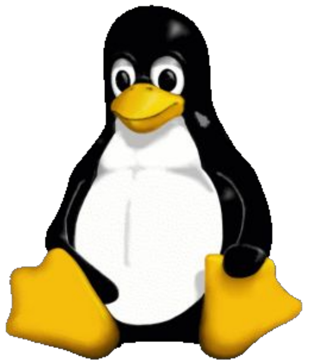
```
# pwru 'dst host 1.1.1.1'
```





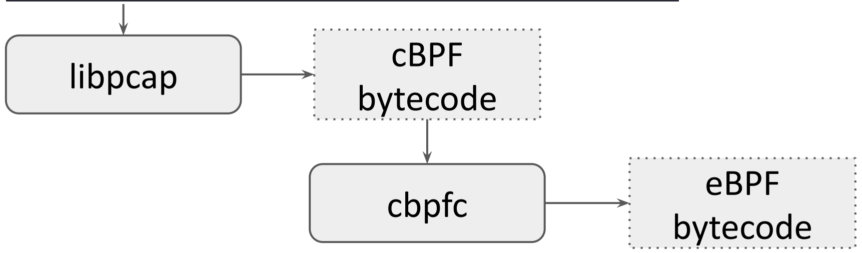
```
clang -target bpf [...]
```

kernel

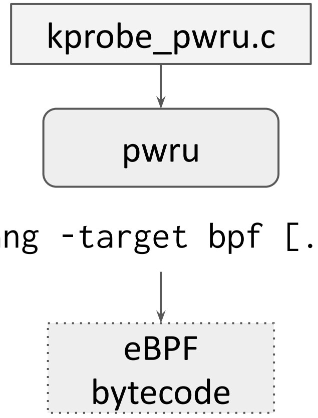


Packet filtering

```
# pwru 'dst host 1.1.1.1'
```

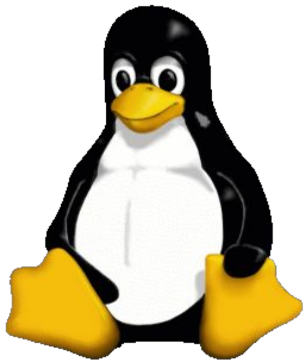


cbpfc by Cloudflare:
cBPF to C or eBPF compiler

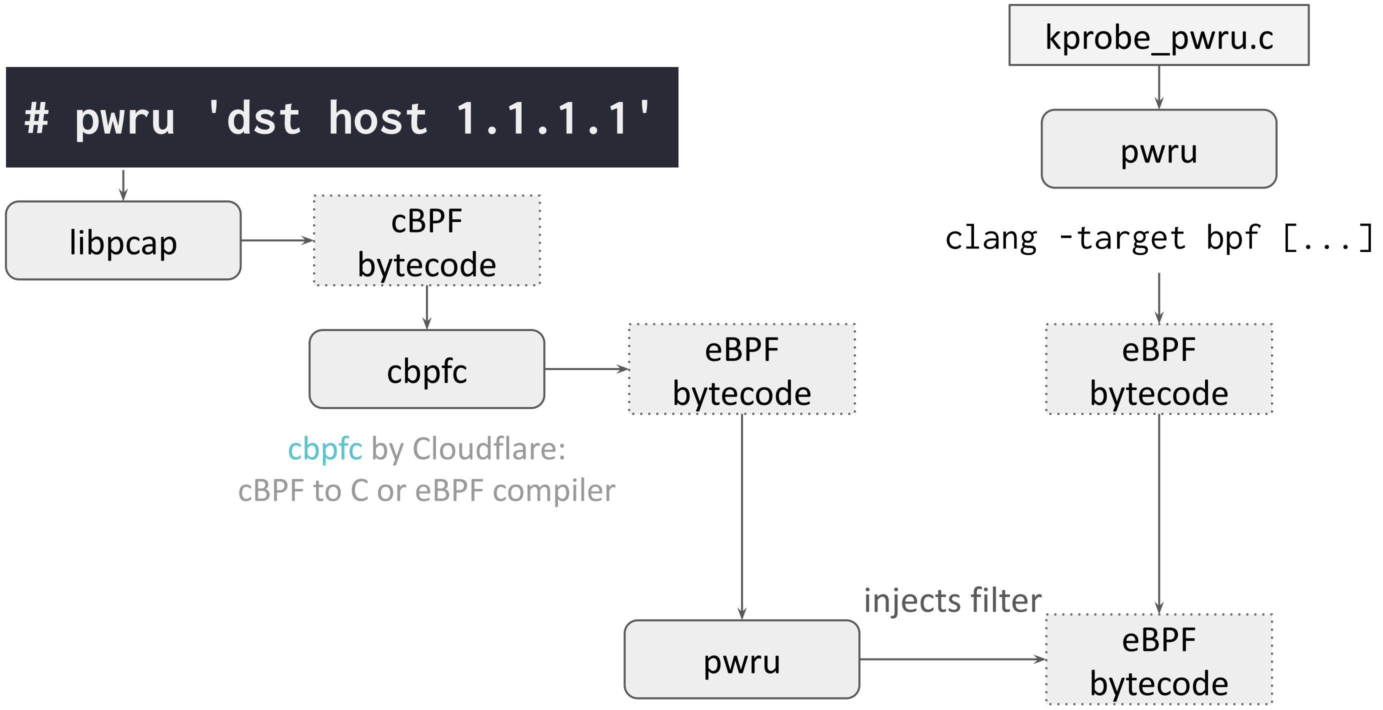


```
clang -target bpf [...]
```

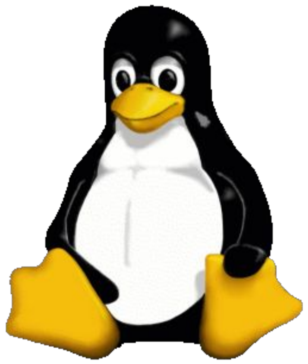
kernel



Packet filtering

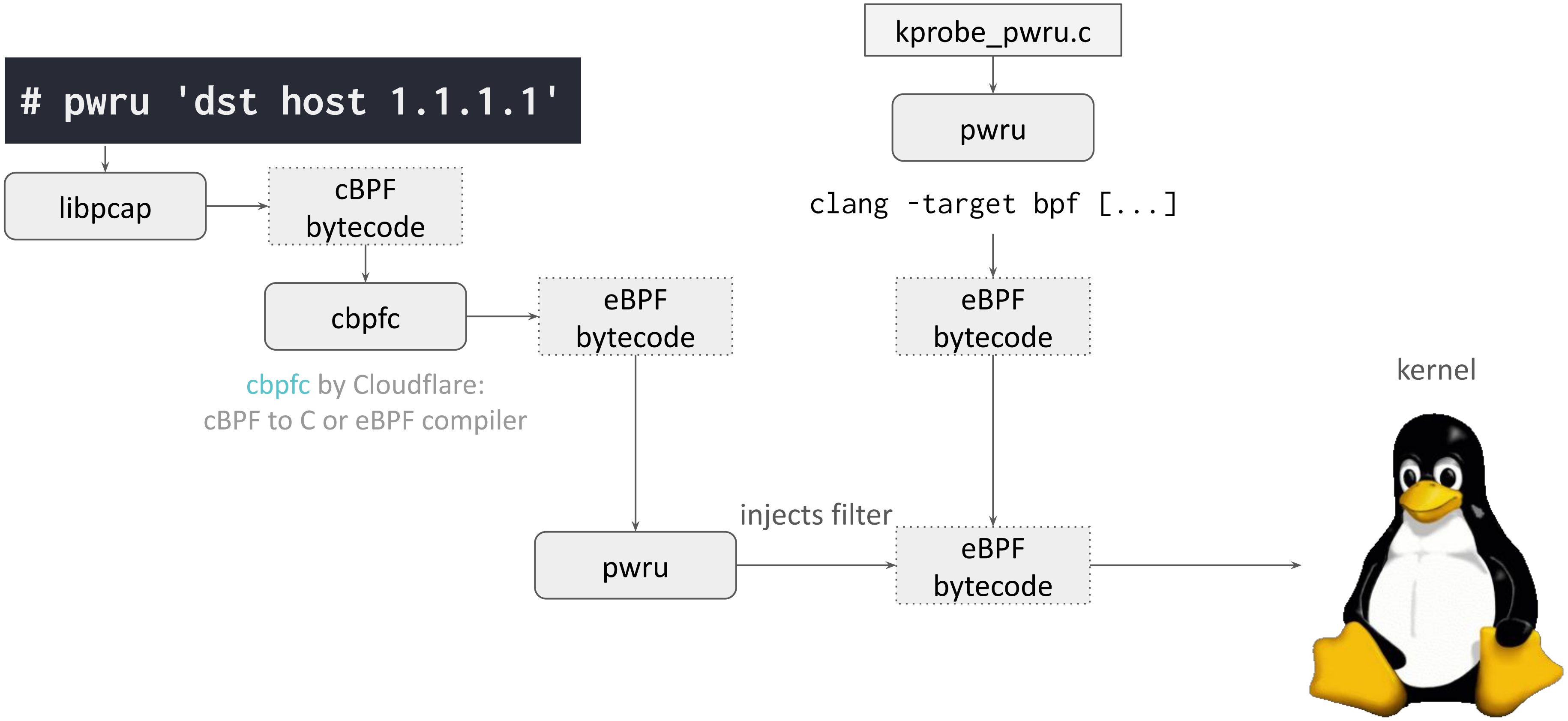


kernel



Packet filtering

```
# pwru 'dst host 1.1.1.1'
```



```
$ pwru --help
Usage: pwru [options] [pcap-filter]
   Available pcap-filter: see "man 7 pcap-filter"
   Available options:
     --all-kmods                    attach to all available kernel modules






     --kmods strings                list of kernel modules names to attach to
```

Trace modules, too

```
$ pwru --help
Usage: pwru [options] [pcap-filter]
   Available pcap-filter: see "man 7 pcap-filter"
   Available options:

     --backend string            Tracing backend('kprobe', 'kprobe-multi'). Will auto-detect if not specified.
```

"Adding new link type BPF_LINK_TYPE_KPROBE_MULTI that attaches kprobe pogram through fprobe API. The fprobe API allows to attach probe on multiple functions at once very fast, because it works on top of ftrace. [...] User provides array of addresses or symbols with count to attach the kprobe program to. The new link_create uapi interface looks like:

```
      struct {
            __u32           flags;
            __u32           cnt;
            __aligned_u64   syms;
            __aligned_u64   addrs;
      }                                                            kprobe_multi;"
```

https://git.kernel.org/torvalds/c/0dcac2725406

Multi-kprobes support

Multi-kprobes support (kernel 5.18+)

```
# pwru --backend kprobe "src host 1.1.1.1" && date
2024/02/02 14:20:51 Attaching kprobes (via kprobe)...
1704 / 1704 [----------------------------------------------------------------------------------------] 100.00% 273 p/s
2024/02/03 14:20:58 Attached (ignored 148)
2024/02/03 14:20:58 Listening for events..
            SKB    CPU          PROCESS                      FUNC
^C2024/02/02 14:21:00 Received signal, exiting program..
2024/02/02 14:21:00 Detaching kprobes...
1556 / 1556 [-----------------------------------------------------------------------------------------] 100.00% 16 p/s
Fri  2 Feb 14:22:37 CET 202
```

Multi-kprobes support (kernel 5.18+)

```
# pwru --backend kprobe "src host 1.1.1.1" && date
2024/02/02 14:20:51 Attaching kprobes (via kprobe)...
1704 / 1704 [---------------------------------------------------------------------------------------] 100.00% 273 p/s
2024/02/03 14:20:58 Attached (ignored 148)
2024/02/03 14:20:58 Listening for events..
            SKB    CPU          PROCESS                      FUNC
^C2024/02/02 14:21:00 Received signal, exiting program..
2024/02/02 14:21:00 Detaching kprobes...
1556 / 1556 [----------------------------------------------------------------------------------------] 100.00% 16 p/s
Fri  2 Feb 14:22:37 CET 202
```

Multi-kprobes support (kernel 5.18+)

```
# pwru --backend kprobe "src host 1.1.1.1" && date
2024/02/02 14:20:51 Attaching kprobes (via kprobe)...
1704 / 1704 [---------------------------------------------------------------------------------------] 100.00% 273 p/s
2024/02/03 14:20:58 Attached (ignored 148)
2024/02/03 14:20:58 Listening for events..
             SKB    CPU          PROCESS                        FUNC
^C2024/02/02 14:21:00 Received signal, exiting program..
2024/02/02 14:21:00 Deta      kprobes...
1556 / 1556 [----------        ----------------------------------------------------------------------] 100.00% 16 p/s
Fri  2 Feb 14:22:37 CET
```

Multi-kprobes support (kernel 5.18+)

```
# pwru --backend kprobe "src host 1.1.1.1" && date
2024/02/02 14:20:51 Attaching kprobes (via kprobe)...
1704 / 1704 [--------------------------------------------------------------------------------------] 100.00% 273 p/s
2024/02/03 14:20:58 Attached (ignored 148)
2024/02/03 14:20:58 Listening for events..
            SKB    CPU          PROCESS                        FUNC
^C2024/02/02 14:21:00 Received signal, exiting program..
2024/02/02 14:21:00 Deta       kprobes...
1556 / 1556 [---------------------------------------------------------------------------------------] 100.00% 16 p/s
Fri  2 Feb 14:22:37 CET

# pwru --backend kprobe-multi "src host 1.1.1.1" && date
2024/02/03 04:20:38 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [----------------------------------------------------------------------------------------] 100.00% ? p/s
2024/02/03 04:20:38 Attached (ignored 0)
2024/02/03 04:20:38 Listening for events..
            SKB    CPU          PROCESS                        FUNC
^C2024/02/03 04:20:40 Received signal, exiting program..
2024/02/03 04:20:40 Detachi     kprobes...
5 / 5 [-----------------------------------------------------------------------------------------------] 100.00% 7 p/s
Sat  3 Feb 04:20:41 CET
```

```
$ pwru --help
Usage: pwru [options] [pcap-filter]
   Available pcap-filter: see "man 7 pcap-filter"
   Available options:
```

```
--filter-netns string        filter netns ("/proc/<pid>/ns/net", "inode:<inode>")
--filter-trace-tc            trace TC bpf progs
--filter-track-skb           trace a packet even if it does not match given filters (e.g., after NAT
                             or tunnel decapsulation)
```

Trace and filter everything you need:

- Filter on network namespaces
- Trace TC programs (XDP: no support yet)
- Track changing SKBs

```
$ pwru --help
Usage: pwru [options] [pcap-filter]
   Available pcap-filter: see "man 7 pcap-filter"
   Available options:
```

```
     --output-meta                print skb metadata
     --output-skb                 print skb
     --output-stack               print stack
     --output-tuple               print L4 tuple
```

Print relevant information

```
# pwru --output-stack "src host 1.1.1.1"
2024/02/03 03:02:17 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [-----------------------------------------------------------------------------------------------] 100.00% 54824 p/s
2024/02/02 13:02:17 Attached (ignored 0)
2024/02/02 13:02:17 Listening for events..
             SKB    CPU          PROCESS                   FUNC
0xffff8d074d8e6f00        2 [irq/172-iwlwifi:queue_4(793)]        inet_gro_receive
inet_gro_receive
napi_gro_receive
ieee80211_rx_napi         [mac80211]
iwl_mvm_pass_packet_to_mac80211 [iwlmvm]
iwl_mvm_rx_mpdu_mq        [iwlmvm]
iwl_mvm_rx_mq_rss         [iwlmvm]
iwl_pcie_rx_handle_rb.constprop.0      [iwlwifi]
iwl_pcie_rx_handle        [iwlwifi]
iwl_pcie_napi_poll_msix [iwlwifi]
__napi_poll
net_rx_action
__do_softirq
do_softirq.part.0
__local_bh_enable_ip
iwl_pcie_irq_rx_msix_handler     [iwlwifi]
irq_thread_fn
irq_thread
kthread
ret_from_fork
ret_from_fork_asm
0xffff8d074d8e6f00        2 [irq/172-iwlwifi:queue_4(793)]        tcp4_gro_receive
tcp4_gro_receive
dev_gro_receive
```

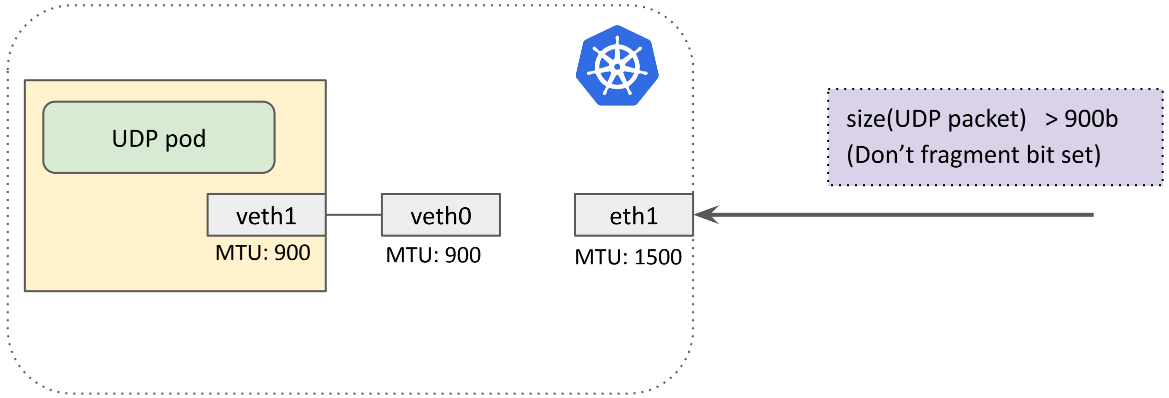
Example: stack trace

```
# pwru --output-stack "src host 1.1.1.1"
2024/02/03 03:02:17 Attaching kprobes (via kprobe-multi)...
1556 / 1556 [------------------------------------------------------------------------------------------------] 100.00% 54824 p/s
2024/02/02 13:02:17 Attached (ignored 0)
2024/02/02 13:02:17 Listening for events..
             SKB    CPU          PROCESS                   FUNC
0xffff8d074d8e6f00        2 [irq/172-iwlwifi:queue_4(793)]        inet_gro_receive
inet_gro_receive
napi_gro_receive
ieee80211_rx_napi         [mac80211]
iwl_mvm_pass_packet_to_mac80211 [iwlmvm]
iwl_mvm_rx_mpdu_mq        [iwlmvm]
iwl_mvm_rx_mq_rss         [iwlmvm]
iwl_pcie_rx_handle_rb.constprop.0     [iwlwifi]
iwl_pcie_rx_handle        [iwlwifi]
iwl_pcie_napi_poll_msix [iwlwifi]
__napi_poll
net_rx_action
__do_softirq
do_softirq.part.0
__local_bh_enable_ip
iwl_pcie_irq_rx_msix_handler     [iwlwifi]
irq_thread_fn
irq_thread
kthread
ret_from_fork
ret_from_fork_asm
0xffff8d074d8e6f00        2 [irq/172-iwlwifi:queue_4(793)]        tcp4_gro_receive
tcp4_gro_receive
dev_gro_receive
```

Example: stack trace

Two examples

Example: MTU misconfiguration



Example: MTU misconfiguration

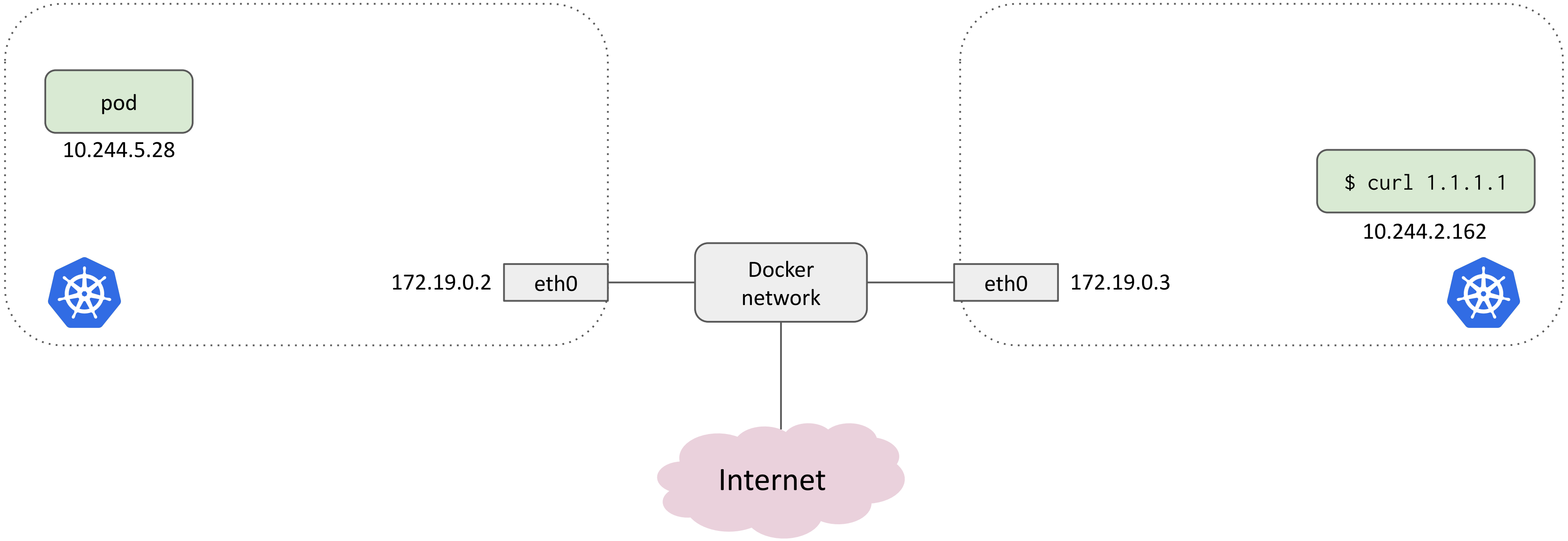
```
# pwru --output-tuple "udp and dst port 443"
[...]
ip_output ifindex=18 mtu=1500, len=1428 sip=192.168.34.11 dip=172.12.0.2
nf_hook_slow ifindex=7 mtu=900, len=1428 sip=192.168.34.11 dip=172.12.0.2
[...]
```

Example: MTU misconfiguration

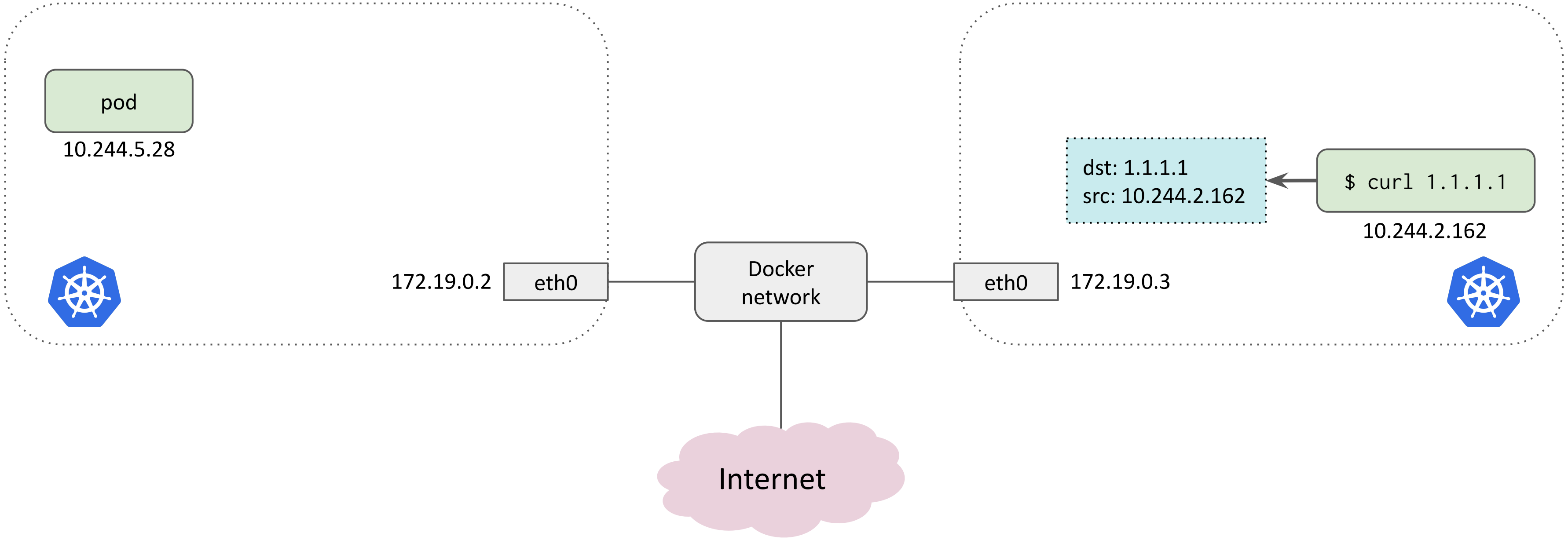
```
# pwru --output-tuple "udp and dst port 443"
[...]
ip_output ifindex=18 mtu=1500, len=1428 sip=192.168.34.11 dip=172.12.0.2
nf_hook_slow ifindex=7 mtu=900, len=1428 sip=192.168.34.11 dip=172.12.0.2
[...]
```

Packet len > MTU

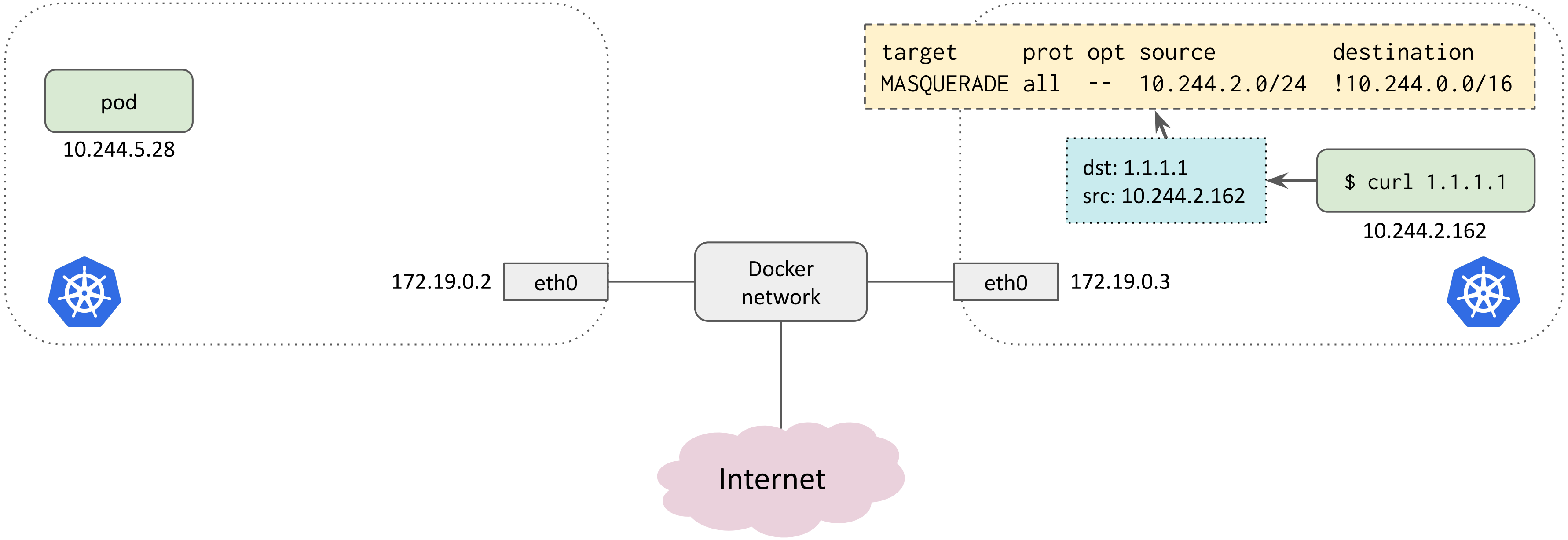
Example: Missing entry in ipset



Example: Missing entry in ipset

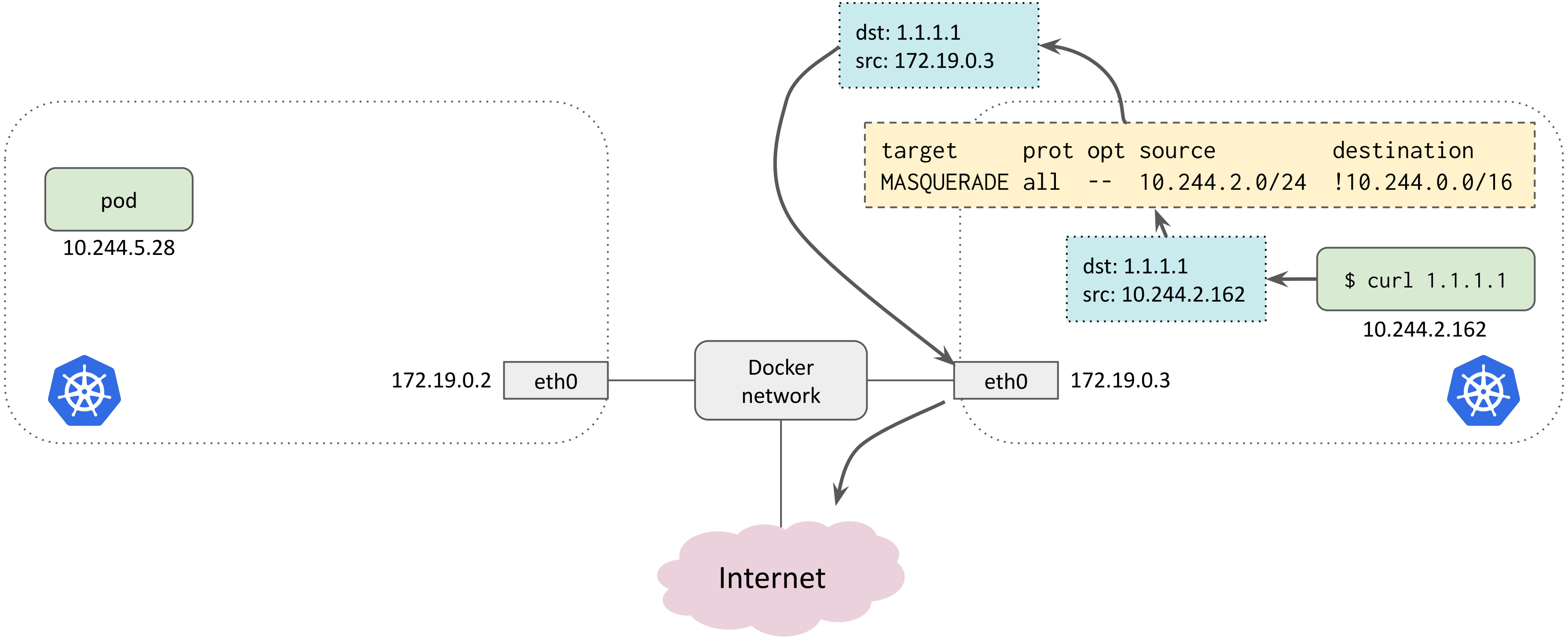


Example: Missing entry in ipset



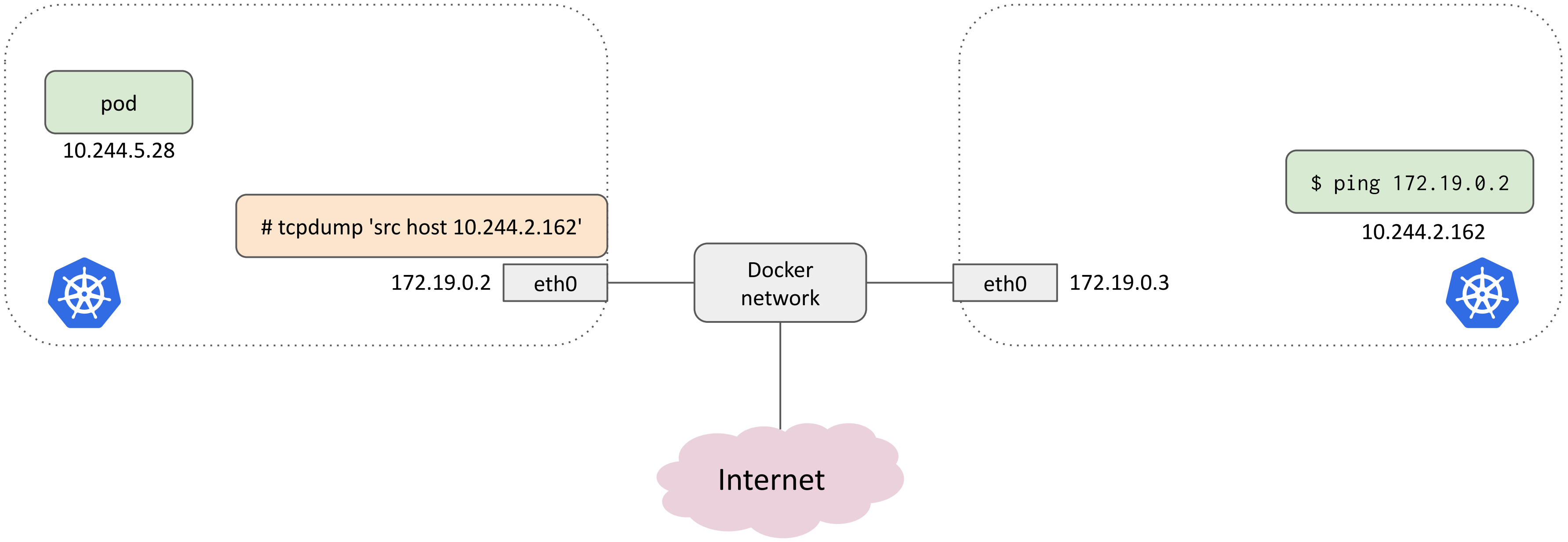
```
target     prot opt source         destination
MASQUERADE all  --  10.244.2.0/24  !10.244.0.0/16
```

Example: Missing entry in ipset



```
target     prot opt source         destination
MASQUERADE all  --  10.244.2.0/24  !10.244.0.0/16
```

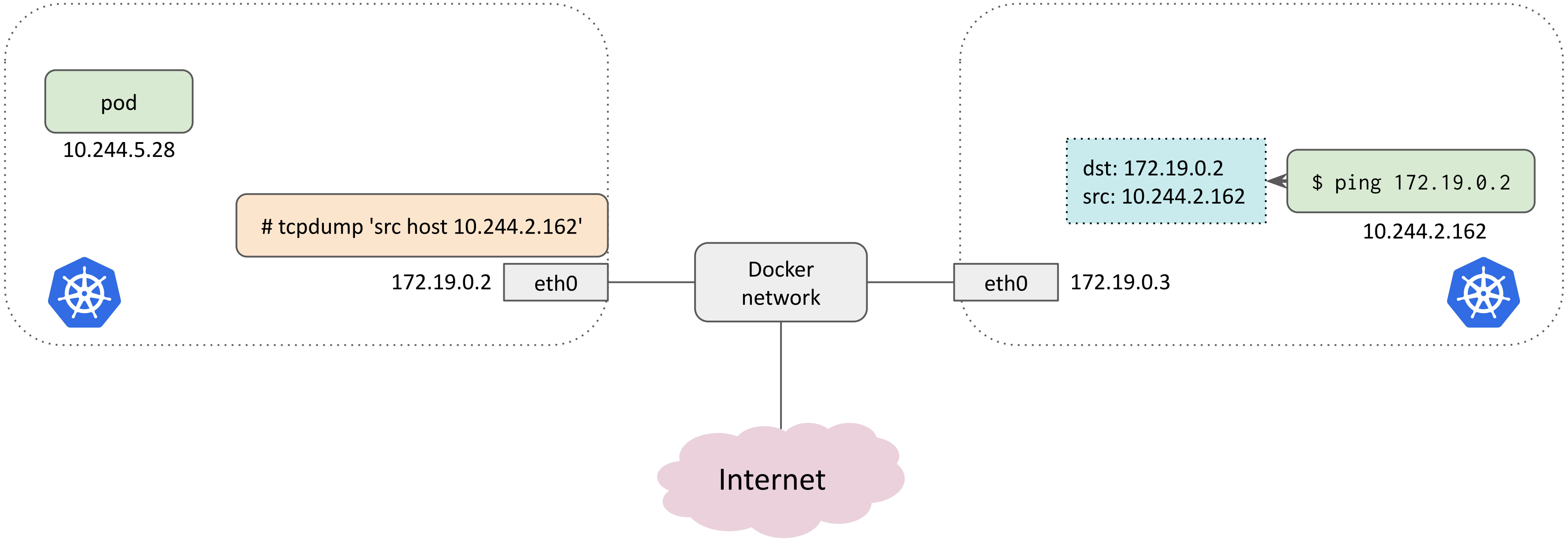
Example: Missing entry in ipset



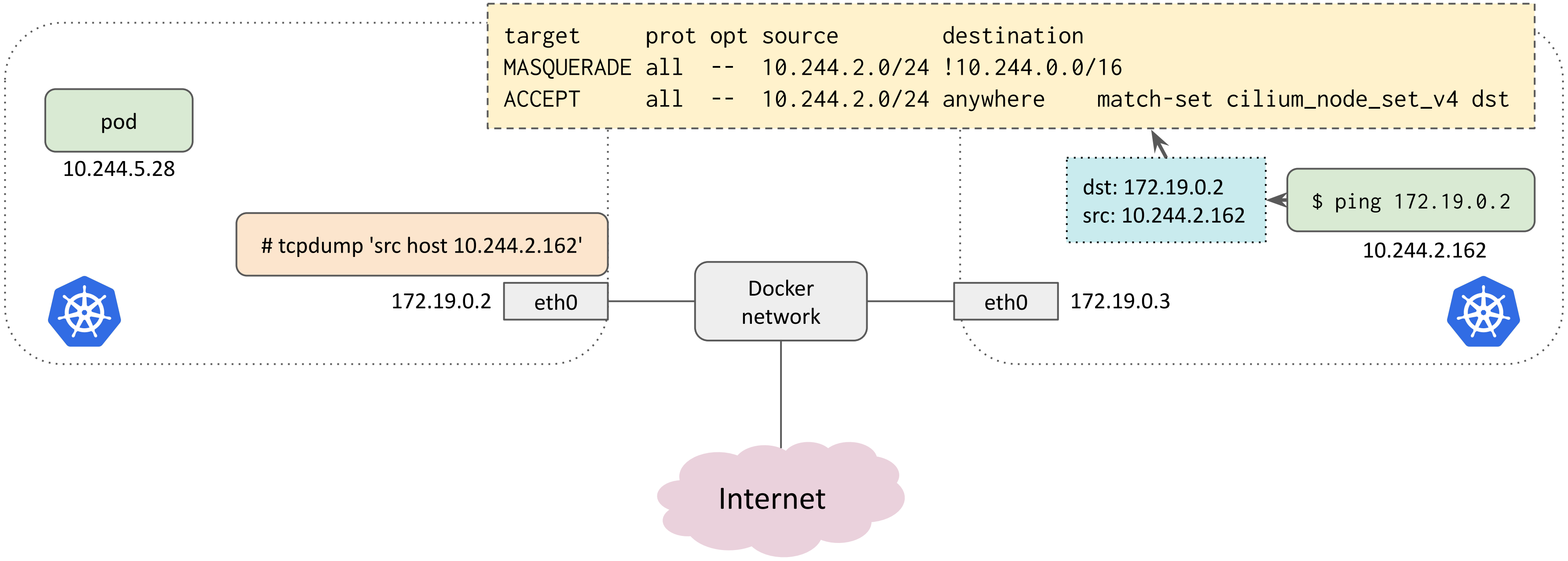
```
# tcpdump 'src host 10.244.2.162'
```

```
$ ping 172.19.0.2
```

Example: Missing entry in ipset

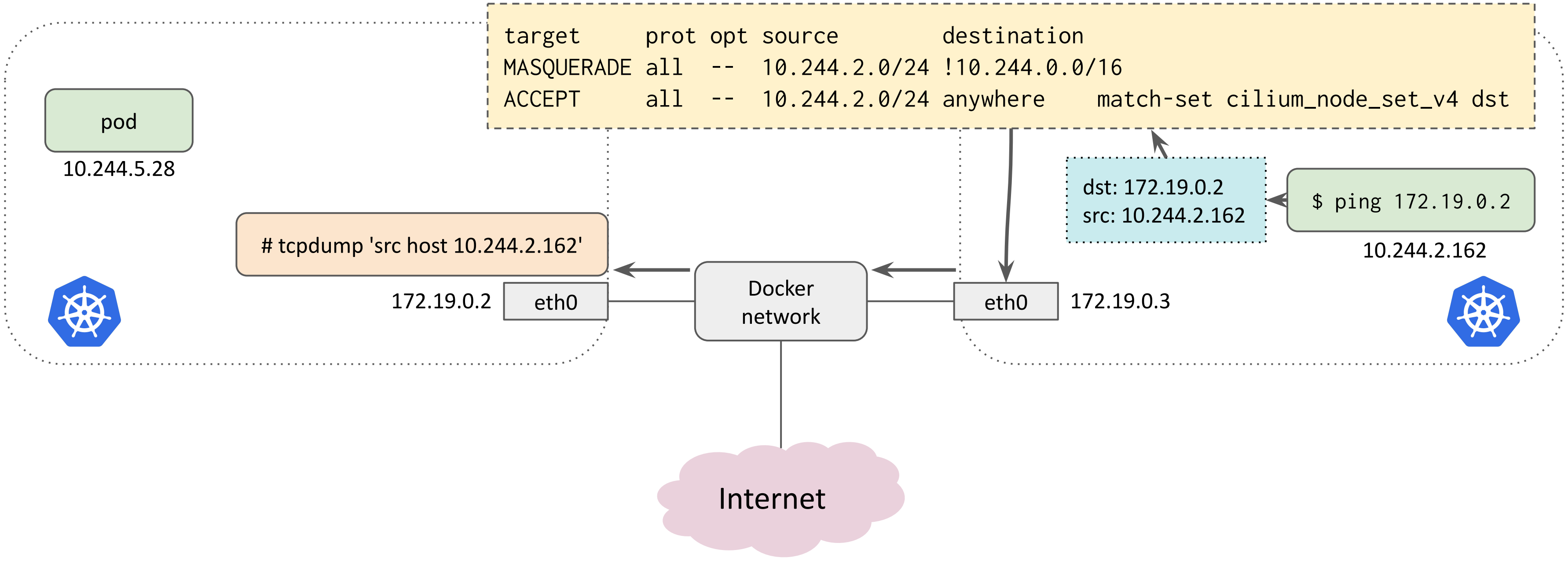


Example: Missing entry in ipset



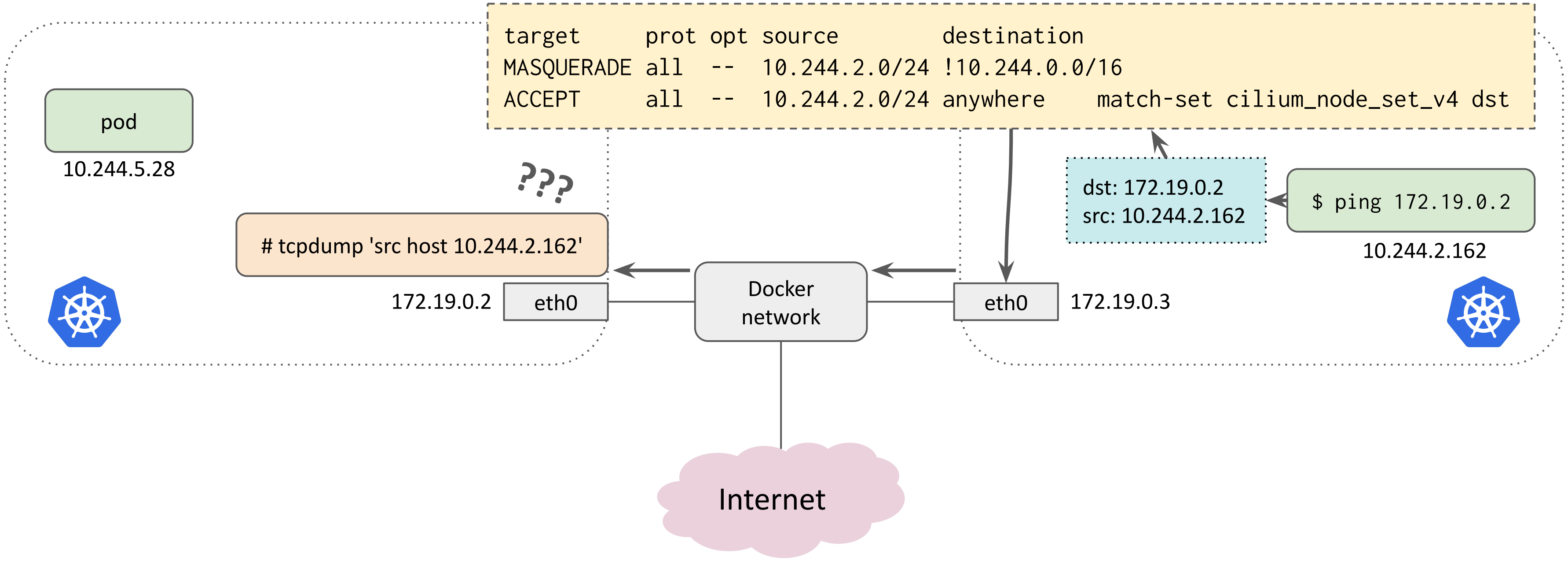
```
target     prot opt source         destination
MASQUERADE all  --  10.244.2.0/24 !10.244.0.0/16
ACCEPT     all  --  10.244.2.0/24 anywhere    match-set cilium_node_set_v4 dst
```

Example: Missing entry in ipset



```
target     prot opt source       destination
MASQUERADE all  --  10.244.2.0/24 !10.244.0.0/16
ACCEPT     all  --  10.244.2.0/24 anywhere    match-set cilium_node_set_v4 dst
```

Example: Missing entry in ipset



```
target     prot opt source        destination
MASQUERADE all  --  10.244.2.0/24 !10.244.0.0/16
ACCEPT     all  --  10.244.2.0/24 anywhere    match-set cilium_node_set_v4 dst
```

Example: Missing entry in ipset

```
# pwru "dst host 172.19.0.2 and icmp"
[...]
                  ip_forward 10.244.2.162:0->172.19.0.2:0(icmp)
                nf_hook_slow 10.244.2.162:0->172.19.0.2:0(icmp)
           ip_forward_finish 10.244.2.162:0->172.19.0.2:0(icmp)
                   ip_output 10.244.2.162:0->172.19.0.2:0(icmp)
                nf_hook_slow 10.244.2.162:0->172.19.0.2:0(icmp)
       apparmor_ip_postroute 10.244.2.162:0->172.19.0.2:0(icmp)
         skb_ensure_writable 10.244.2.162:0->172.19.0.2:0(icmp)
         skb_ensure_writable 10.244.2.162:0->172.19.0.2:0(icmp)
    inet_proto_csum_replace4 10.244.2.162:0->172.19.0.2:0(icmp)
       __xfrm_decode_session 172.19.0.3:0->172.19.0.2:0(icmp)
             decode_session4 172.19.0.3:0->172.19.0.2:0(icmp)
security_xfrm_decode_session 172.19.0.3:0->172.19.0.2:0(icmp)
```

Masqueraded when nf_hook_slow() returns

Example: Missing entry in ipset

```
Chain CILIUM_POST_nat (1 references)
target     prot opt source               destination
ACCEPT     all  --  10.244.2.0/24        anywhere             match-set cilium_node_set_v4 dst
                                          /* exclude traffic to cluster nodes from masquerade */
MASQUERADE  all  --  10.244.2.0/24      !10.244.0.0/16
                                                               /* cilium masquerade non-cluster */
ACCEPT     all  --  anywhere             anywhere             mark match 0xa00/0xe00
                                              /* exclude proxy return traffic from masquerade */
SNAT       all  --  localhost            anywhere
                           /* cilium host->cluster from 127.0.0.1 masquerade */ to:10.244.2.205
```

Example: Missing entry in ipset

```
Chain CILIUM_POST_nat (1 references)
target     prot opt source               destination
ACCEPT     all  --  10.244.2.0/24        anywhere             match-set cilium_node_set_v4 dst
                                          /* exclude traffic to cluster nodes from masquerade */
MASQUERADE  all  --  10.244.2.0/24       !10.244.0.0/16
                                                           /* cilium masquerade non-cluster */
ACCEPT     all  --  anywhere             anywhere             mark match 0xa00/0xe00
                                              /* exclude proxy return traffic from masquerade */
SNAT       all  --  localhost            anywhere
                           /* cilium host->cluster from 127.0.0.1 masquerade */ to:10.244.2.205


# ipset list
cilium_node_set_v4:
Number of entries: 2
Members:
172.19.0.5
172.19.0.3
```

No 172.19.0.2

pwru in brief

- eBPF-based tool to debug packet trajectories in the Linux kernel networking stack
- Hooks on kernel functions processing SKBs
- Picks up things where tpcdump leaves them
- Supports pcap-filter syntax, several additional filters
- Traces TC programs; traces kernel module functions; tracks modified SKBs
- Prints packet-level metadata, call stack, full SKB, …
- Ideal for troubleshooting complex networking issue in the Linux kernel

Note: Other tools using many k(ret)probes

retsnoop: debug kernel, mainly by retrieving return values from functions

ipftrace2: trace packets, similar to pwru, some features differ

Tetragon: security events detection; motivation for multi-(k|u)probes

Credits

- Aditi Ghag & Martynas Pumpitis
  Beyond printf and tcpdump: Debugging Kubernetes Networking with eBPF
  (KubeCon NA 2021)

- The pwru contributors ❤

Further reading

- *Going from Packet Where Aren't You to pwru* (Cilium blog)

Thank you!

github.com/cilium/pwru

Contributions welcome!

cilium

ISOVALENT

https://fosdem.org/2024/schedule/event/fosdem-2024-3425-packet-where-are-you-track-in-the-stack-with-pwru/