Demystifying eBPF

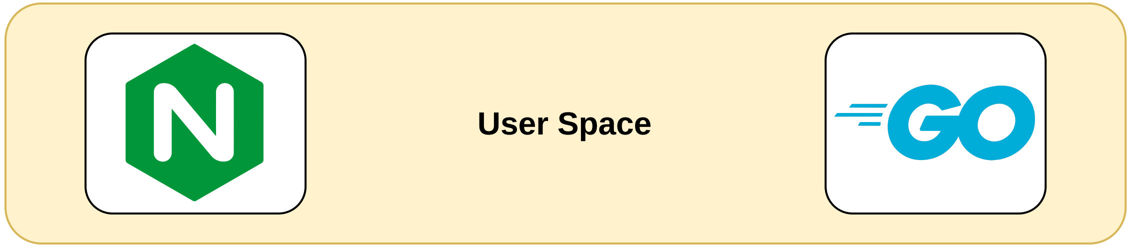
eBPF Firewall from scratch

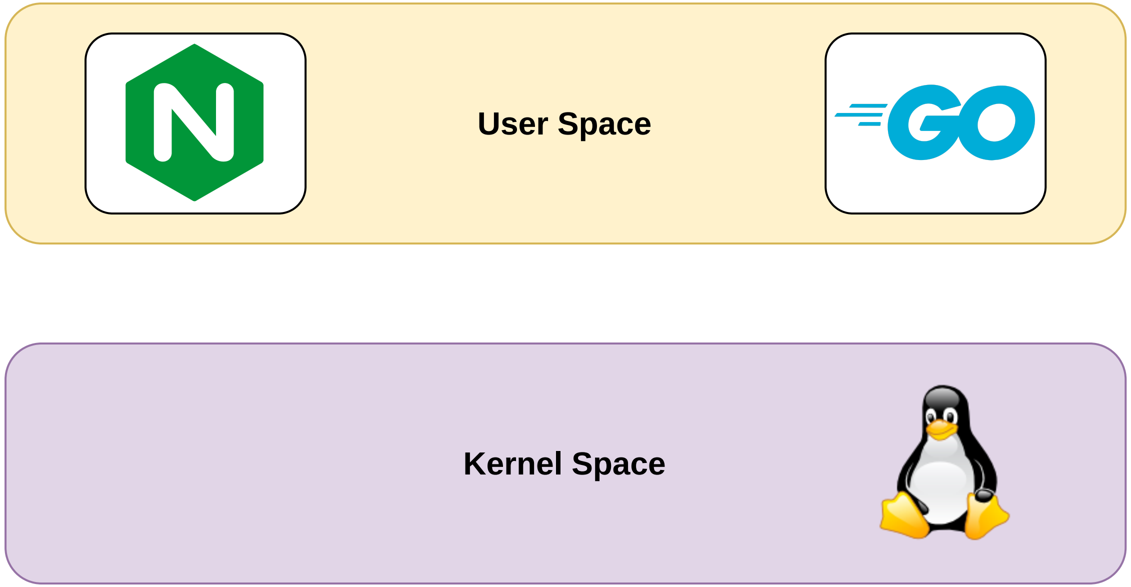
Filip Nikolic

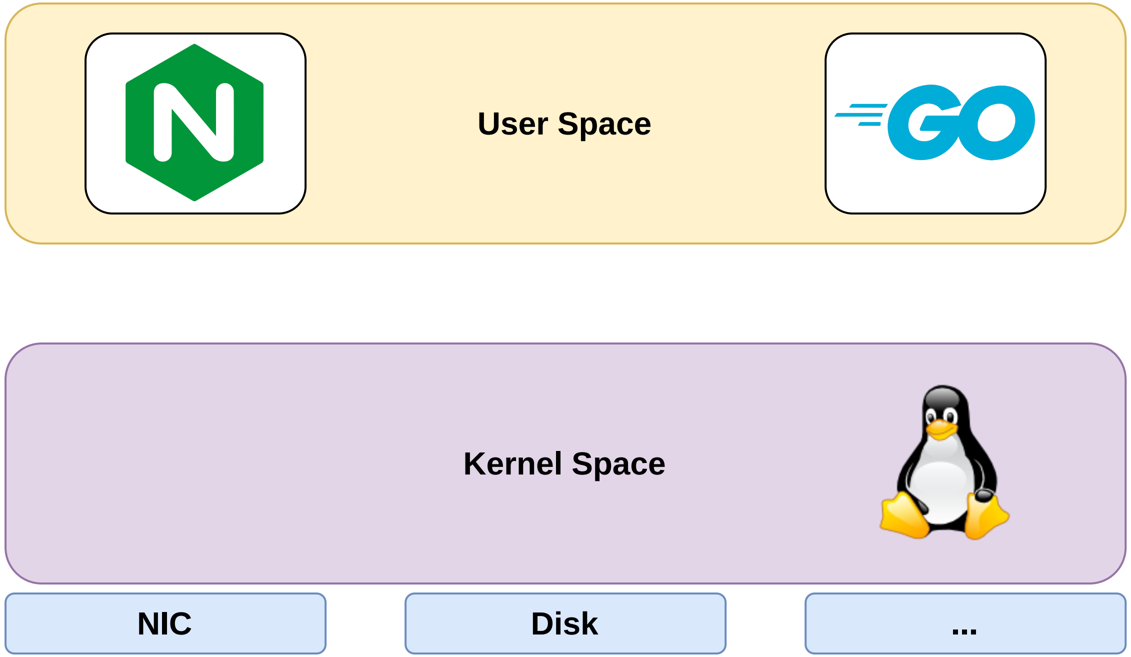
Kubernetes Engineer at PostFinance

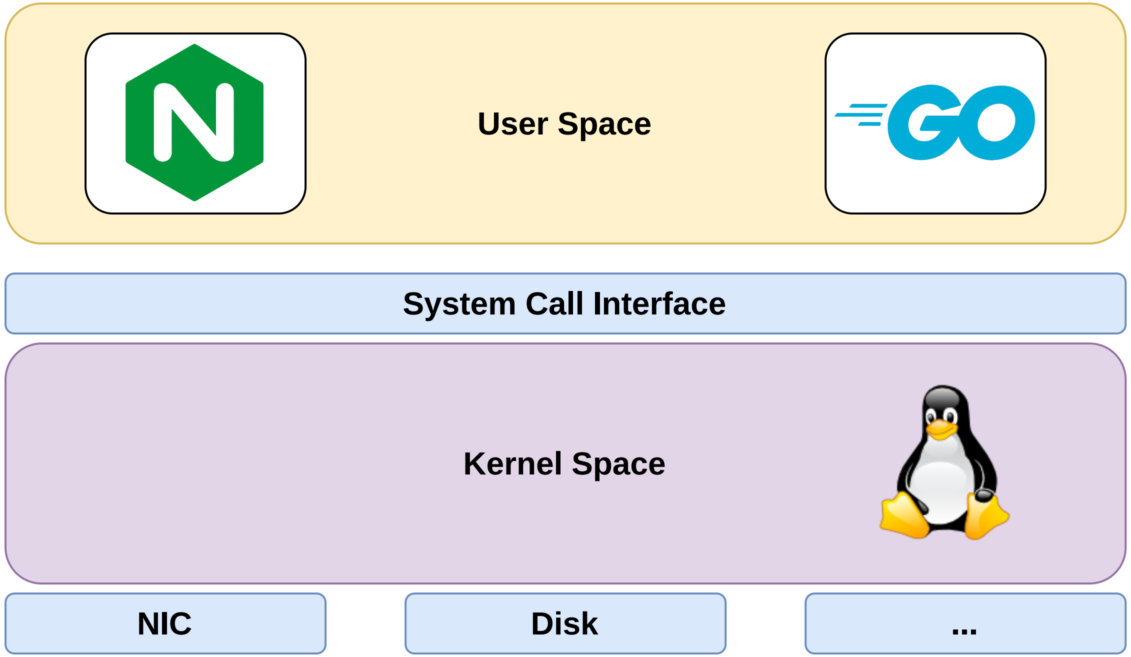
CNCF Projects

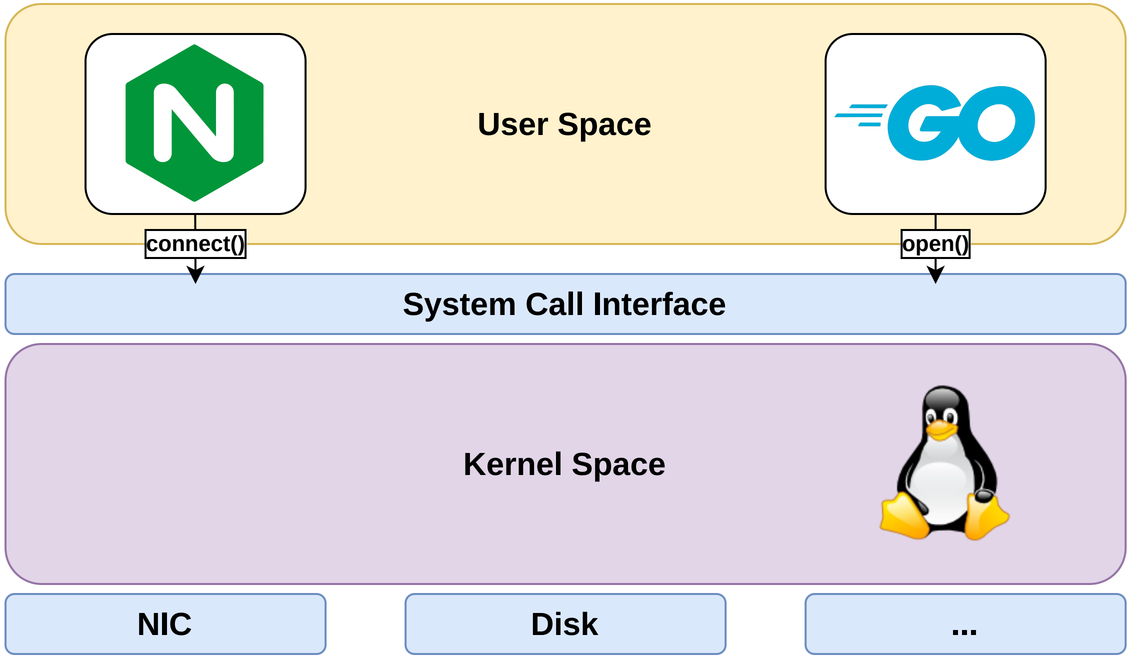
- Falco
  - Security
- Pixie
  - Observability
- Cilium
  - Networking

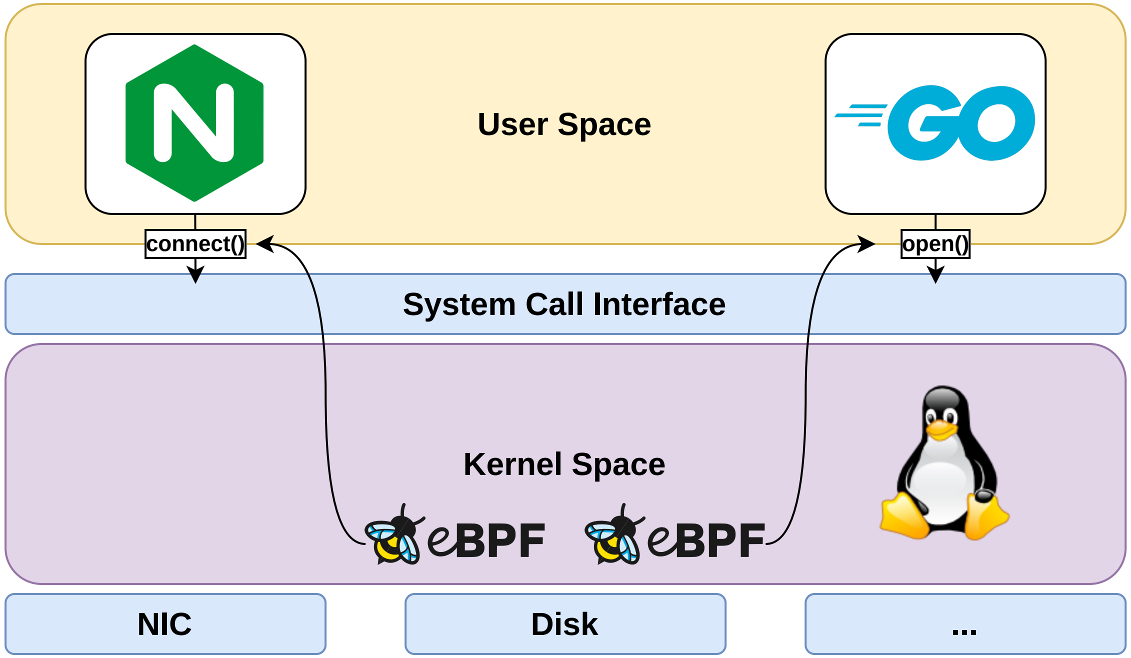

User Space

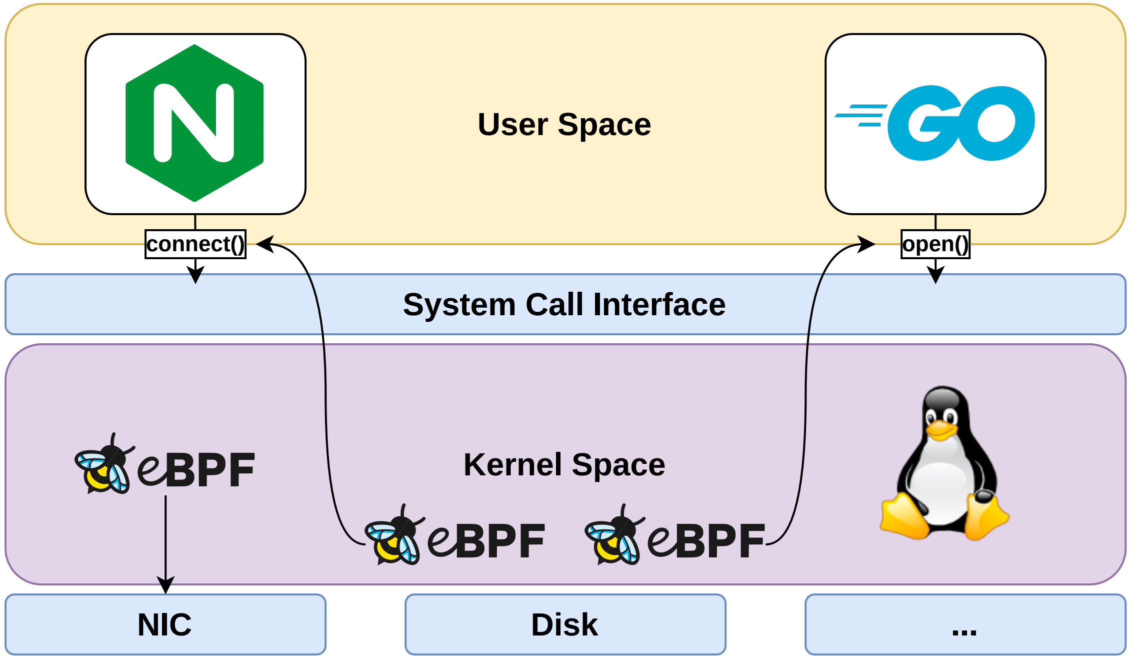

User Space
GO
Kernel Space

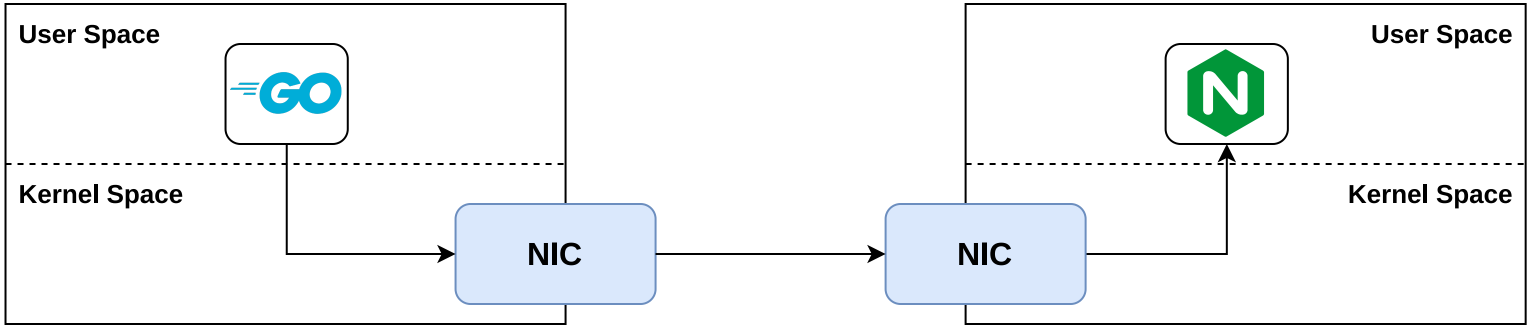

User Space
Kernel Space
NIC
Disk
...


User Space
System Call Interface
Kernel Space
NIC
Disk
...


User Space
connect()
open()
System Call Interface
Kernel Space
NIC
Disk
...


User Space
connect()
open()
System Call Interface
Kernel Space
eBPF
eBPF
NIC
Disk
...


User Space
connect()
open()
System Call Interface
eBPF
Kernel Space
eBPF
eBPF
NIC
Disk
...


User Space
Kernel Space
NIC
NIC
User Space
Kernel Space

Pass everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // pass all packets
  return XDP_PASS;
}
```

Pass everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // pass all packets
  return XDP_PASS;
}
```

Pass everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // pass all packets
  return XDP_PASS;
}
```

Pass everything

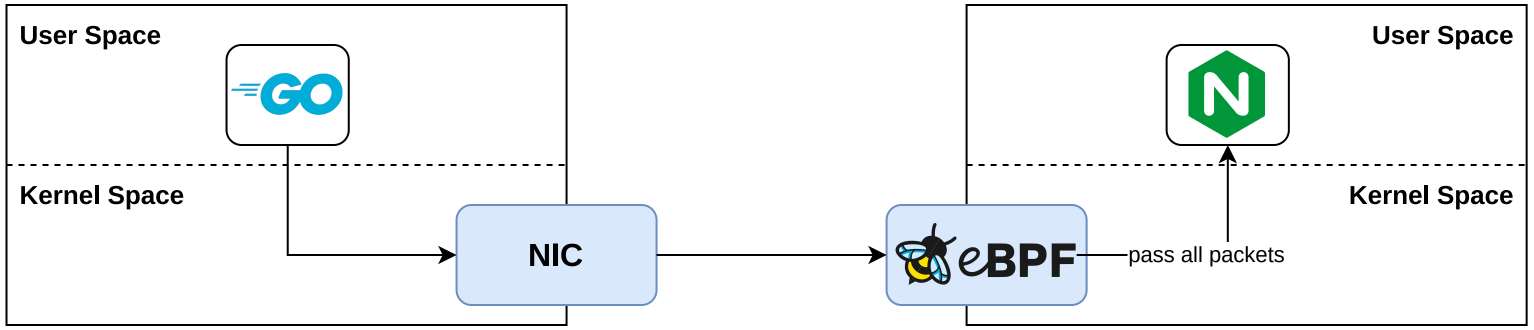
```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // pass all packets
  return XDP_PASS;
}
```

Pass everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // pass all packets
  return XDP_PASS;
}
```

Pass everything

```
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // pass all packets
  return XDP_PASS;
}
```



Drop everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // drop all packets
  return XDP_DROP;
}
```

Drop everything

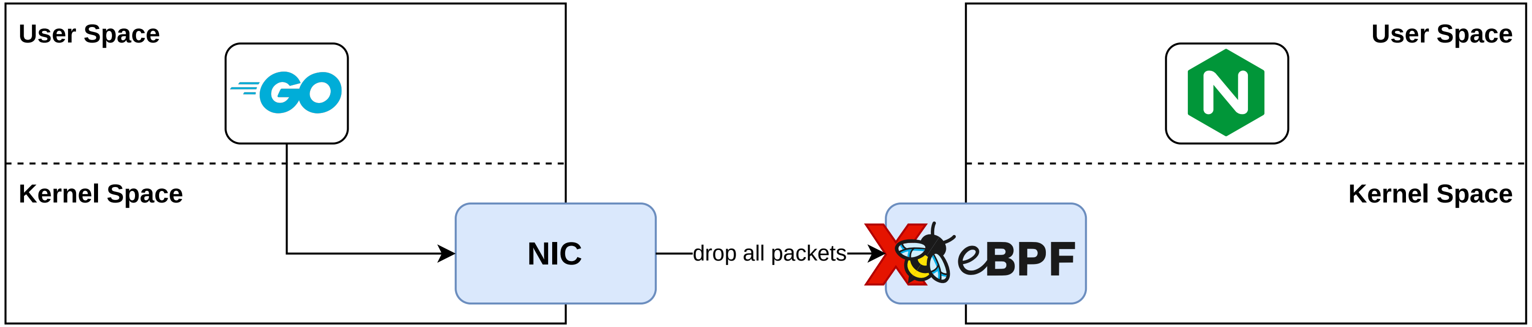
```
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // drop all packets
  return XDP_DROP;
}
```

Drop everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // drop all packets
  return XDP_DROP;
}
```

Drop everything

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // drop all packets
  return XDP_DROP;
}
```



Pseudocode (what we want)

```
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Pseudocode (what we want)

```
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Pseudocode (what we want)

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Pseudocode (what we want)

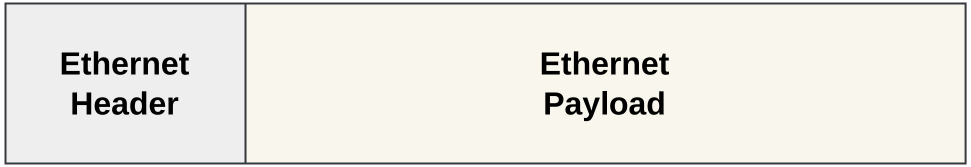
```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Pseudocode (what we want)

```
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

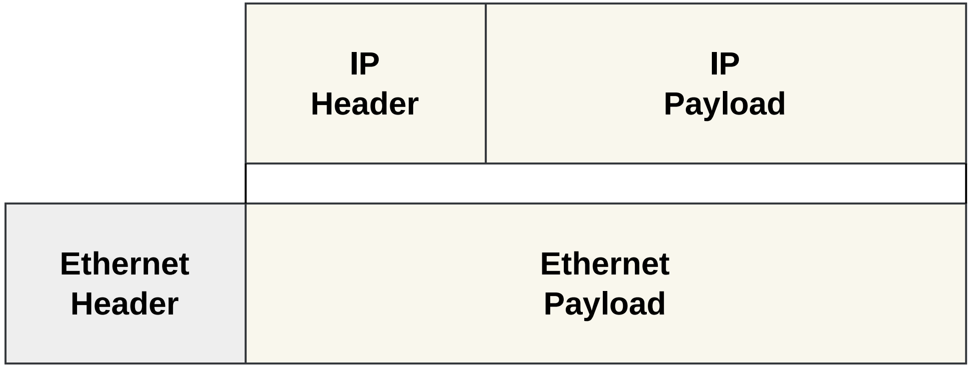
Pseudocode (what we want)

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```



Pseudocode (what we want)

```
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  protocol = find_out_protocol;

  // drop all packets of type unwanted protocol
  if (protocol == UNWANTED_PROTOCOL)
    return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```



Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

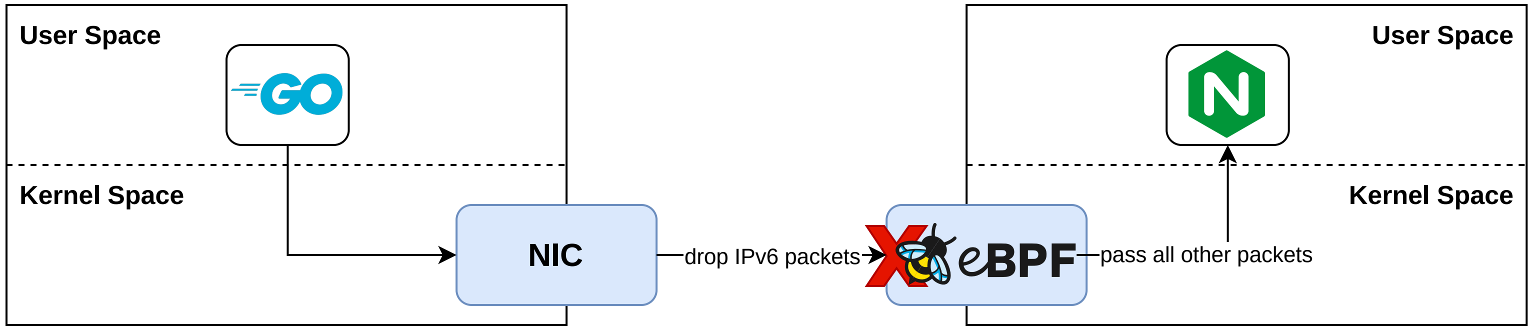
Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Drop IPv6

```c
SEC("xdp_kcd_zurich")
int xdp_kcd_zurich_firewall(struct xdp_md *ctx)
{
  // define variables that represent the packet
  void *packet_start = (void *)(long)ctx->data;
  void *packet_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = packet_start;

  // satisfy eBPF verifier
  if (packet_start + sizeof(*eth) > packet_end)
    return XDP_DROP;

  // find out the next protocol
  __u16 protocol = eth->h_proto;

  // drop all IPv6 packets
  if (protocol == bpf_htons(ETH_P_IPV6))
      return XDP_DROP;

  // pass all other packets
  return XDP_PASS;
}
```

Outcome


User Space
Kernel Space
NIC
drop IPv6 packets
eBPF
pass all other packets
User Space
Kernel Space

Summary

- Event-driven
- Versatile
- Fast
- Secure
- Increasingly more popular

Questions ?