



第二届 eBPF 开发者大会

www.ebpftravel.com

XDP ACL 在 Shopee 虚拟网络网关上的实战经验分享

黄富, 13 Apr. 2024

中国·西安



第二届 eBPF 开发者大会

www.ebpftravel.com

AGENDA

- 1 虚拟网络与 XDP 网关简介
- 2 XDP ACL 背景简介
- 3 XDP ACL 方案研究
- 4 XDP ACL 方案实现讲解
- 5 XDP ACL 方案落地效果

中国·西安

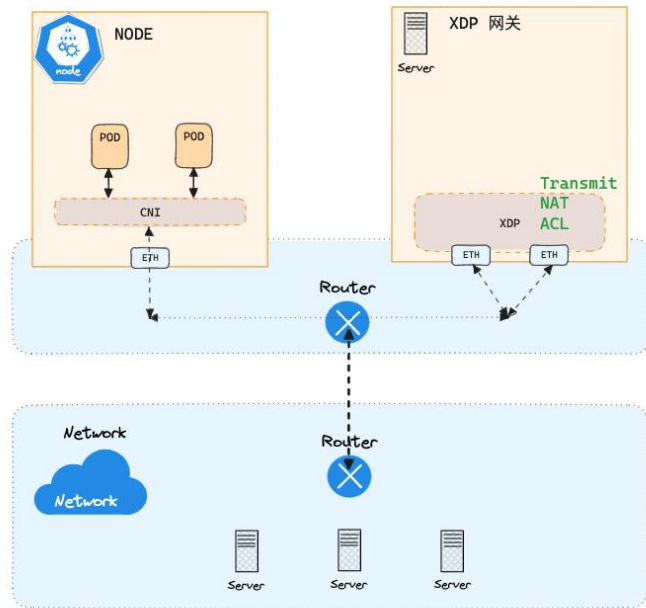
虚拟网络与 XDP 网关简介

虚拟网络与 XDP 网关简介

虚拟网络网关：是一种用于连接虚拟网络与外部网络的设备或服务。

在 **Shopee**，基于 **XDP** 实现的虚拟网络网关提供了如下能力：

- 跨网络通信：允许虚拟网络终端（e.g. Kubernetes pod）和物理网络终端（e.g. baremetal server）之间通信。
- 网络地址转换：将私有 IP 地址转换为公网 IP 地址，提供面向公网的网络通信能力。
- 安全性功能：主要提供访问控制列表（ACL）的能力，保护集群中的数据和资源安全。

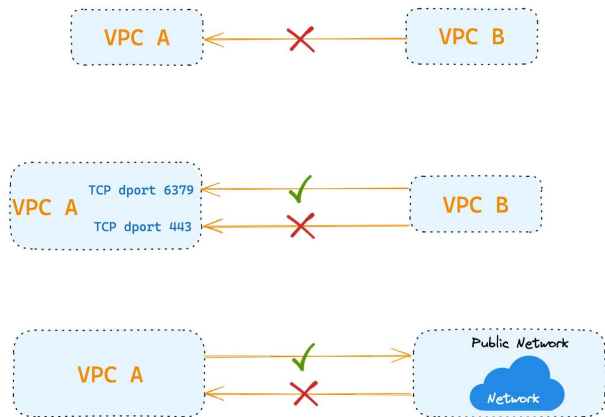


XDP ACL 背景简介

XDP ACL 背景简介

在 **Shopee**，虚拟网络的用户有不少 **VPC** 维度的 **ACL** 需求，比如：

1. **VPC A** 拒绝来自 **VPC B** 的流量访问
2. **VPC A** 只对外提供提供 **TCP dport 6379** 的 **Redis** 服务
3. **VPC A** 允许访问公网，但不允许公网的流量主动访问 **VPC A**



XDP ACL 方案研究

XDP ACL 方案研究

XDP ACL 相关论文:

- [eBPF / XDP based firewall and packet filtering](#)
- [Securing Linux with a Faster and Scalable Iptables](#)
- [TupleMerge: Fast Software Packet Processing for Online Packet Classification](#)

XDP ACL 相关博客文章:


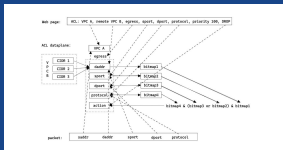
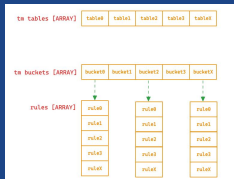
- [eBPF技术实践: 高性能ACL](#)
- [eBPF Talk: 再论高性能 eBPF ACL](#)
- [eBPF Talk: 低性能 eBPF ACL](#)

XDP ACL 相关开源项目:

- [GitHub hi-glenn/xdp_acl](#)
- [GitHub Asphaltt/xdp_acl](#)
- [\[RFC PATCH\] bpf: introduce new bpf map type BPF_MAP_TYPE_WILDCARD](#)

XDP ACL 方案研究

对比 3 个 XDP ACL 方案:

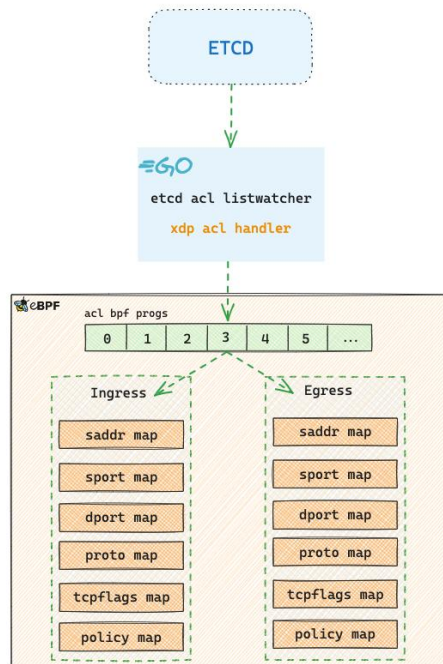
	iptables-like approach	bitmap approach	TupleMerge approach
匹配方式	<ul style="list-style-type: none"> One by one 	<ul style="list-style-type: none"> Tuple match 	<ul style="list-style-type: none"> Merged Space Search 
实现难度	<ul style="list-style-type: none"> 低 	<ul style="list-style-type: none"> 高 	<ul style="list-style-type: none"> 高
支持的规则复杂度	<ul style="list-style-type: none"> 低 	<ul style="list-style-type: none"> 高 	<ul style="list-style-type: none"> 低
千条规则匹配耗时	<ul style="list-style-type: none"> 高 	<ul style="list-style-type: none"> 低 	<ul style="list-style-type: none"> 低

XDP ACL 方案研究

因为 **bitmap** 方案满足高 **ACL** 规则复杂度、低匹配耗时的需求，所以需要在 **XDP** 网关项目中做 **POC**，确认能够在 **XDP** 网关项目中落地。

确认点：

1. **ACL** 规则能从 **ETCD** 下发到 **XDP bpf maps**。
2. **XDP bpf maps** 正确保存 **ACL** 规则的数据。
3. 网络包能匹配到 **ACL** 规则。



XDP ACL 方案实现讲解

XDP ACL 方案实现讲解

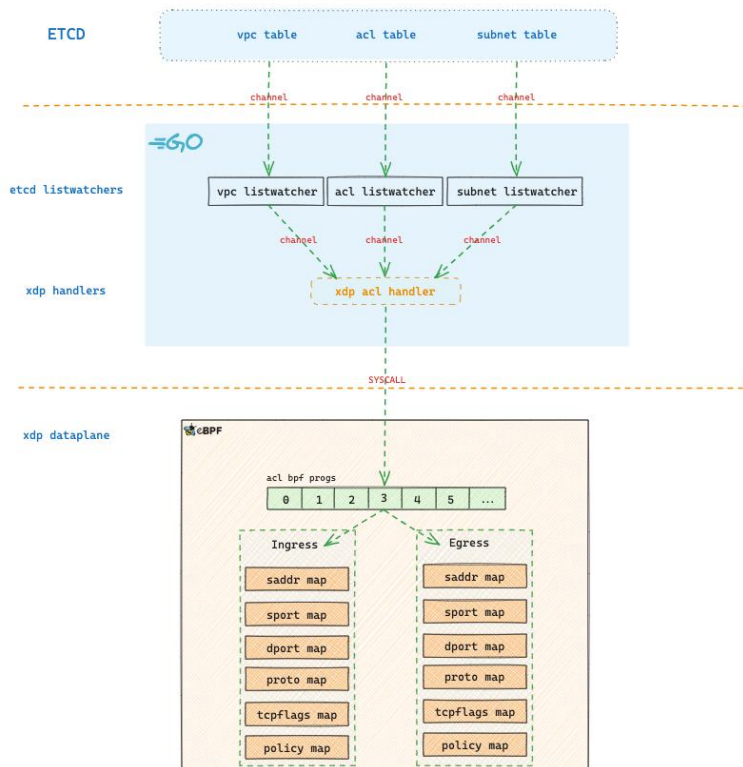
基于 **bitmap** 的 **XDP ACL** 方案比较复杂，实现难度较高，难点较多：

XDP ACL

- Match ACL rules for one direction
- Find ACL policy for the matched bitmaps
- Match ACL rules for two directions

Go acl module

- Aggregate data from ETCD
- Sort ACL rules by priority
- Prepare data to save to bpf maps
- Update acl PROG_ARRAY



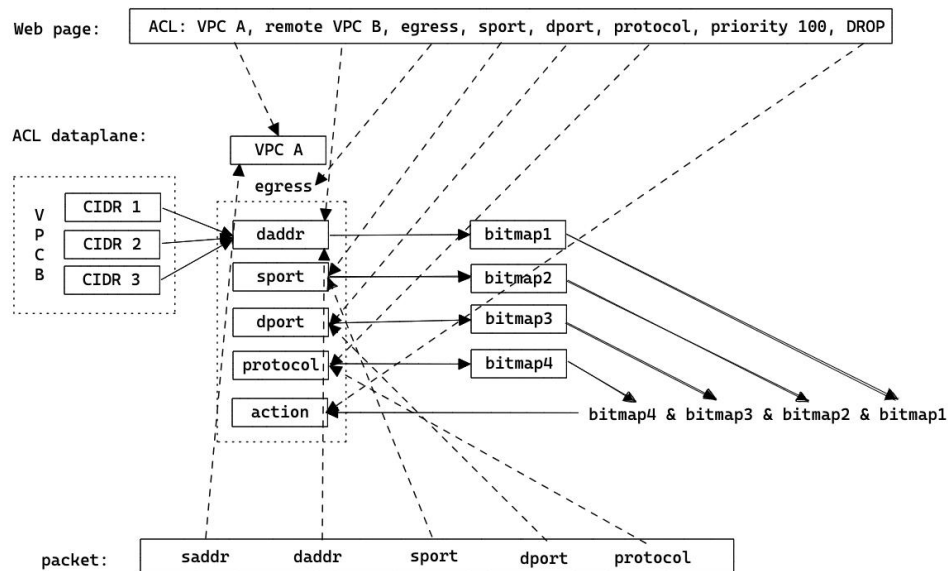
XDP ACL 方案实现讲解: Match ACL rules for one direction

对于一个网络包，有 2 个方向的规则需要匹配：

通过目的 IP 地址可以查找到 VPC B，然后去匹配 VPC B 的 INGRESS 方向的 ACL 规则；

通过源 IP 地址可以查找到 VPC A，然后去匹配 VPC A 的 EGRESS 方向的 ACL 规则。

匹配 VPC A EGRESS 方向的 ACL 规则的示例：



XDP ACL 方案实现讲解: Match ACL rules for one direction

对于一个网络包, 有 2 个方向的规则需要匹配:

通过目的 IP 地址可以查找到 VPC B, 然后去匹配 VPC B 的 INGRESS 方向的 ACL 规则;

通过源 IP 地址可以查找到 VPC A, 然后去匹配 VPC A 的 EGRESS 方向的 ACL 规则。

匹配 VPC A EGRESS 方向的 ACL 规则的示例:

VPC A 的 bpf prog 的 EGRESS 方向有 4 个 bpf map:

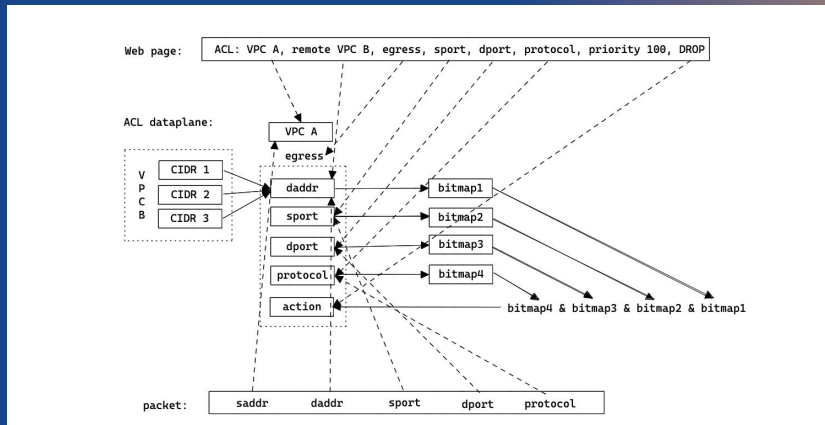
1. **daddr:** lpm trie map, 保存 VPC B 的所有 CIDR
2. **sport:** hash map, 保存所有 ACL 规则里的 sport
3. **dport:** hash map, 保存所有 ACL 规则里的 dport
4. **protocol:** array map, 保存所有 ACL 规则里的 protocol 信息
5. **action:** array map, 保存所有 ACL 规则里的 action 信息

其中, **daddr**、**sport**、**dport**、**protocol** 等 bpf map 的 value 是 **bitmap**。

网络包匹配 ACL 规则的过程:

1. 通过 **saddr** 查找到 VPC A
2. 通过 **tailcall** 运行 VPC A 的 **acl** bpf prog, 并指定匹配 EGRESS 方向的规则
3. 通过 **daddr** 在 **daddr** map 中查找, 得到 **bitmap1**
4. 通过 **sport** 在 **sport** map 中查找, 得到 **bitmap2**
5. 通过 **dport** 在 **dport** map 中查找, 得到 **bitmap3**
6. 通过 **protocol** 在 **protocol** map 中查找, 得到 **bitmap4**
7. 将得到的 4 个 **bitmap** 进行按位与操作, 得到 **bitmap5**
8. 计算 **bitmap5** 中第一个值为 1 的比特的索引
9. 使用该索引到 **action** map 中查找, 得到最终的 **action**

网络包匹配 INGRESS 方向的 ACL 规则的过程与此类似。



XDP ACL 方案实现讲解: Find ACL policy for the matched bitmaps

bitmap: 是一个 uint64 数组，每一个比特位都对应一条 ACL 规则。

数组之间如何进行按位与操作？

同时遍历 bitmap1, bitmap2, bitmap3, bitmap4, 然后对遍历中的 4 个 uint64 进行按位与操作，保存到 bitmap5。把 bitmap1 当作 bitmap5 使用，就不需要额外的 bitmap 了。

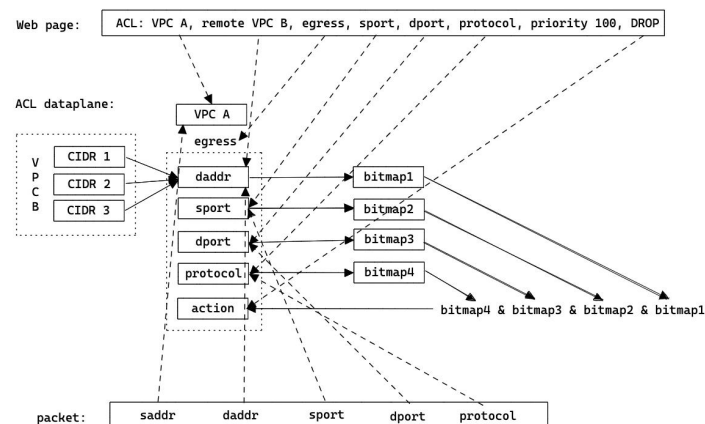
如何计算 bitmap 中第一个值为 1 的比特的索引？

遍历该 bitmap，如果第 n 个 uint64 不为 0，则索引计算方式：

$(n \ll 6) + \text{ffs64}(\text{bitmap}[n])$

ffs64() 的实现方式，请参考 [内核 ffs\(\)](#) 函数的实现。

计算出该索引后，即可到 **action map** 中查找对应 ACL 规则的 action 信息。



XDP ACL 方案实现讲解: Match ACL rules for two directions

网络包匹配 VPC A EGRESS 方向 ACL 规则的过程:

1. 通过 saddr 查找到 VPC A
2. 通过 **tailcall** 运行 VPC A 的 acl bpf prog, 并指定匹配 EGRESS 方向的规则
3. ...

网络包匹配 VPC B INGRESS 方向 ACL 规则的过程:

1. 通过 daddr 查找到 VPC B
2. 通过 **tailcall** 运行 VPC B 的 acl bpf prog, 并指定匹配 INGRESS 方向的规则
3. ...

tailcall 的限制: bpf prog1 tailcall bpf prog2 后, 不会返回 bpf prog1 了。

问题是: 如何使用 **tailcall** 又能拿到 2 个方向的结果呢?

答案请看右边的代码片段:

1. **__noinline**: 使用 tailcall in bpf2bpf 特性。
2. **volatile**: 禁止 clang 编译器对 `acl_action()` 函数返回值的优化处理。
3. **xdp**: 使用 XDP metadata 特性传递方向信息。

```
static __noinline int
acl_action(struct xdp_md *xdp, u32 progid)
{
    volatile int ret = ACL_ACTION_NOT_MATCHED;

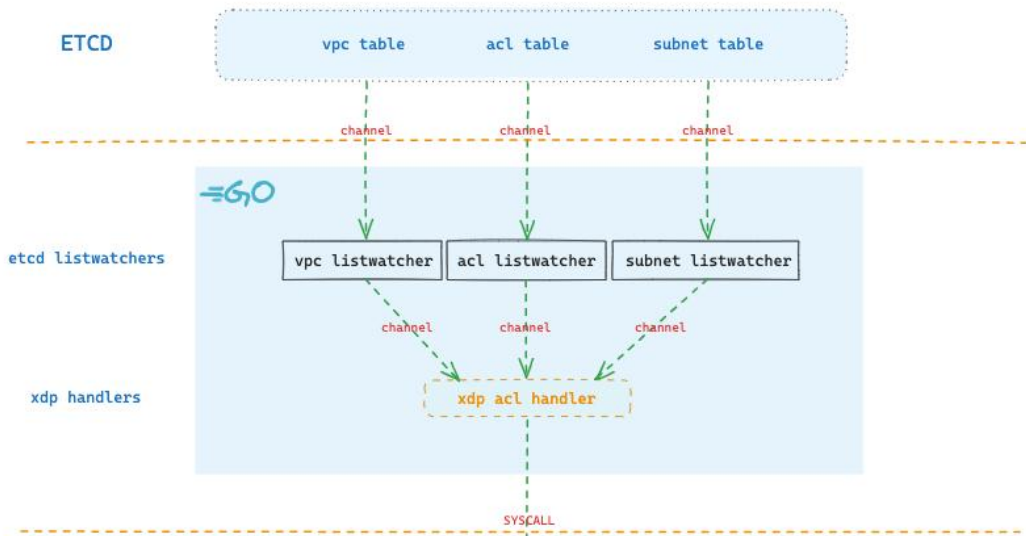
    bpf_tail_call(xdp, &acl_progs, progid);

    return ret;
}
```


XDP ACL 方案实现讲解: Aggregate data from ETCD

在 Go acl module 里, 需要聚合 vpc, acl, subnet 信息:

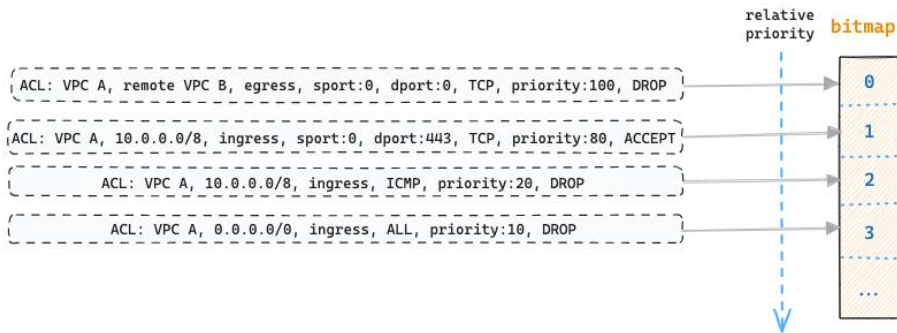
1. 将 vpc 和 acl 规则关联起来。
2. 每个 vpc 有其唯一的 ID, 用来当作 acl PROG_ARRAY 的索引。
3. 如果 acl 规则里的目标地址是另一个 vpc, 则需要将该 vpc 展开成它的 subnet 列表。



XDP ACL 方案实现讲解: Sort ACL rules by priority

在 **Go acl module** 里, 需要根据 **priority** 排序 ACL 规则:

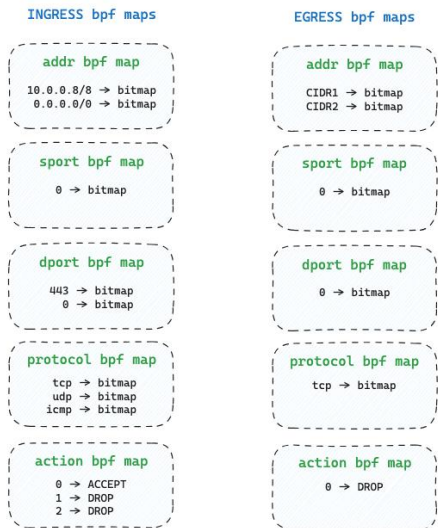
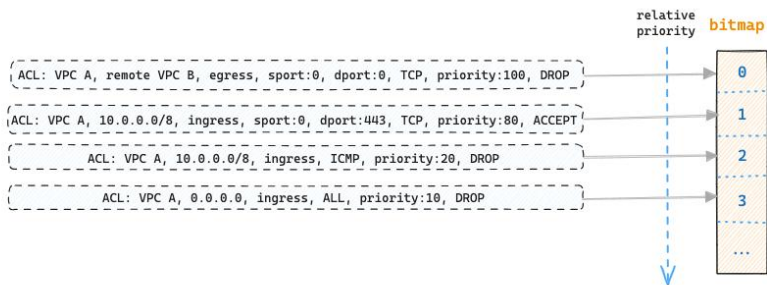
1. 根据 **priority** 以及其它字段进行排序。
2. 排序是为了确定 ACL 规则在 **bitmap** 中的相对位置: 在 **bitmap** 中的索引越小代表优先级越高。



XDP ACL 方案实现讲解: Prepare data to save to bpf maps

在 **Go acl module** 里, 将排序后的 ACL 规则信息保存到 **bpf maps** 里:

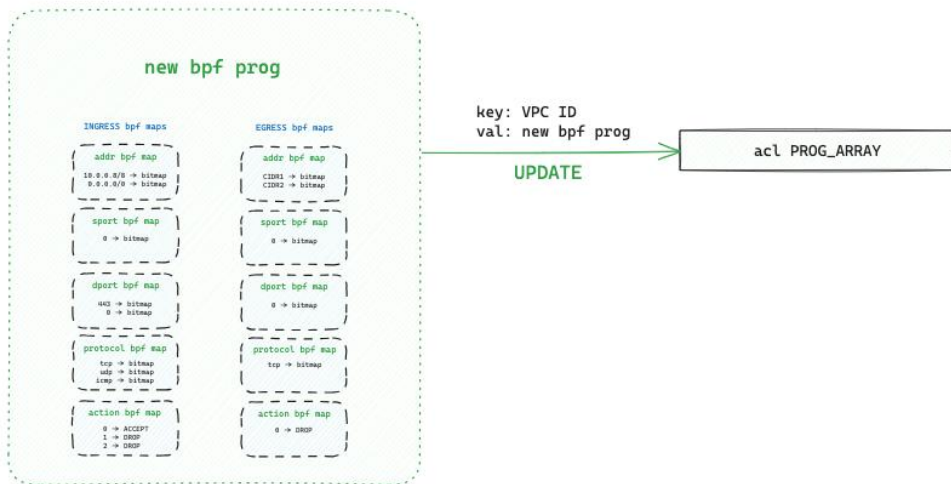
1. 将排序后的 ACL 规则的索引按照 **addr, sport, dport, protocol** 等维度保存到 **bitmap** 中。
2. 按方向将 **addr, sport, dport, protocol** 的 **bitmap** 保存到对应的 **bpf map** 中。
3. 以 ACL 规则的索引为 **key**, ACL 规则 **action** 为 **value** 保存到 **action bpf map** 中。



XDP ACL 方案实现讲解: Update acl PROG_ARRAY

在 **Go acl module** 里，将排序后的 **ACL** 规则信息都保存到 **bpf maps** 后，更新 **acl PROG_ARRAY** 中对应的 **bpf prog**:

1. 每个 VPC 有个唯一的 ID 充当 **acl PROG_ARRAY** 的索引。
2. 每次更新 **ACL** 规则时，使用一个全新的 bpf prog，并将 **ACL** 规则信息保存到新的 bpf maps 中。
3. 以 VPC ID 为索引，更新 **acl PROG_ARRAY**。



XDP ACL 方案落地效果



XDP ACL 方案落地效果

支持配置比较复杂的 **ACL** 规则：

如下规则，不需要拆分成：

1. 80 + 10.1.1.0/24 + 0
2. 8080 + 10.1.1.0/24 + 0
3. 443 + 10.1.1.0/24 + 0
4. ...

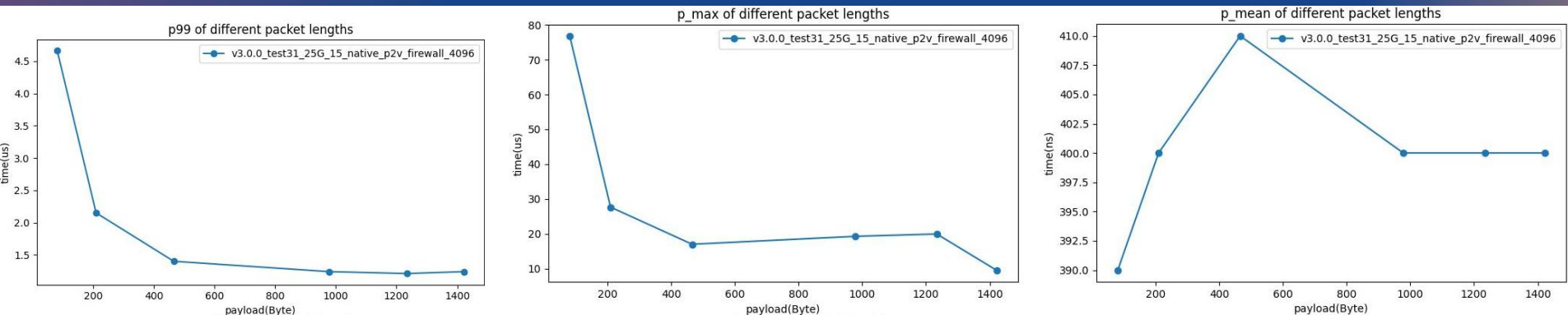
* ACL Rules:

Priority	Direction	Protocol	Port	Remote	Remote Port	Filter Type	Action
100	INGRESS	TCP	80 8080 443 6379	10.1.1.0/24 10.1.2.0/24 10.1.3.0/24	0	<input type="radio"/> Accept <input checked="" type="radio"/> Drop	 

XDP ACL 方案落地效果

配置 4096 条 ACL 规则，不同 payload 长度的每次匹配 ACL 规则的耗时：

环境：25G Mellanox NIC x2





第二届 **eBPF**开发者大会

www.ebpftravel.com

Q&A