(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 116488945 B
(45)授权公告日 2023.09.15

(21)申请号 202310733631.3
(22)申请日 2023.06.20
(65)同一申请的已公布的文献号
申请公布号 CN 116488945 A
(43)申请公布日 2023.07.25
(73)专利权人 杭州默安科技有限公司
地址 310000 浙江省杭州市余杭区仓前街道余杭塘路2616号3号楼1楼
(72)发明人 郑天驰 高小龙 李天华
(74)专利代理机构 杭州裕阳联合专利代理有限公司 33289
专利代理师 田金霞
(51)Int.Cl.
H04L 9/40(2022.01)
G06F 16/22(2019.01)
G06F 16/2455(2019.01)

(56)对比文件
CN 109996307 A,2019.07.09
US 2023104129 A1,2023.04.06
CN 115580497 A,2023.01.06
CN 115658220 A,2023.01.31
CN 105100038 A,2015.11.25
CN 115913778 A,2023.04.04
CN 116226855 A,2023.06.06
CN 114338405 A,2022.04.12
CN 115622748 A,2023.01.17
CN 113608824 A,2021.11.05
EP 4160408 A1,2023.04.05
毕小红;刘渊;陈飞.微服务应用平台的网络性能研究与优化.计算机工程.2017,(05),全文.

审查员 张改红

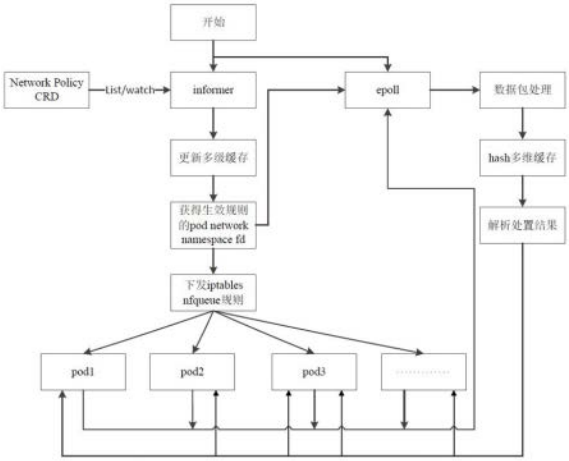
权利要求书2页 说明书6页 附图1页

(54)发明名称

一种容器网络隔离方法和系统

(57)摘要

本发明公开一种容器网络隔离方法和系统，方法包括:创建k8s下的自定义Network Policy资源，填写包括生效的k8s kind、namespace和name，以实现对对应的pod生效;在用户态程序中，监控Network Policy资源变化，下发iptables nfqueue规则到生效的pod内核网络协议栈中;所述iptables nfqueue规则添加在流量出栈后的第一个位置，用以拦截从容器内的应用程序发出的数据包;在用户态程序中，获取拦截的数据包，完成分析后将分析后的处置结果返回nfqueue。本发明使用iptables中的nfqueue，将内核态的握手包流量导入用户态中，根据重新定义的Network Policy自定义资源，实现对数据包的处置。



CN 116488945 B

1.一种容器网络隔离方法,其特征在于,包括以下步骤:

创建k8s下的自定义Network Policy资源,填写包括生效的k8s kind、namespace和name,以实现对对应的pod生效,无需填写label;

在用户态程序中,监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中;

所述iptables nfqueue规则添加在nffilter raw表OUTPUT chain,用以拦截从容器内的应用程序发出的数据包;

在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue。

2. 根据权利要求1所述的一种容器网络隔离方法,其特征在于,所述监控Network Policy资源变化,包括以下步骤:

监控 Network Policy的变化,当Network Policy发生变化时,获取更新的 Network Policy资源,将当前接收的Network Policy资源转换成作用的源地址IP、目的地址IP,以及目的地址端口,写入本地缓存。

3. 根据权利要求1所述的一种容器网络隔离方法,其特征在于,所述监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中的方法包括:

获取Network Policy中填写的需要生效的pod容器的pid,计算获得network namespace fd,

根据network namespace fd将iptables nfqueue规则下发到对应pod容器的network namespace中。

4. 根据权利要求1所述的一种容器网络隔离方法,其特征在于,还包括:通过epoll统一管理所有pod,将所有生效了规则的pod的network namespace fd加入epoll中,

epoll监控到事件后,读取pod中出栈数据包。

5.根据权利要求2所述的一种容器网络隔离方法,其特征在于,所述本地缓存为hash多维缓存,

通过所述hash多维缓存储包括:源地址IP、目的地址IP,以及携带优先级的策略列表,与拦截的数据包提取的源地址IP、目的地址IP,以及目的地址端口具有映射关系,不同目的地址端口对应不同的策略。

6.根据权利要求5所述的一种容器网络隔离方法,其特征在于,在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue的方法包括:

响应于流量出栈后的第一个位置发生新事件,拦截数据包,并分析所述数据包携带的包括源地址IP、目的地址IP,以及目的地址端口信息,映射到hash多维缓存后,找到对应的策略,判断如何处置。

7.一种容器网络隔离系统,其特征在于,包括以下结构:

网络策略单元,用于k8s下的Network Policy资源,填写生效的k8s kind、namespace和name,以实现对对应的pod生效,无需填写label;

策略监控单元,用于在用户态程序中,监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中;

所述iptables nfqueue规则添加在nffilter raw表OUTPUT chain,用以拦截从容器内

的应用程序发出的数据包；

事件监控处置单元，用于在用户态程序中，获取拦截的数据包，完成分析后将分析后的处置结果返回nfqueue。

8.根据权利要求7所述的一种容器网络隔离系统，其特征在于，还包括hash多维缓存单元，通讯连接用户态程序；

所述hash多维缓存单元包括一维存储模块，二维存储模块和三维存储模块，

一维存储模块的hash key包括源地址IP，二维存储模块的hash key包括目的地址IP，三维存储模块包括携带优先级的策略列表，不同目的地址端口对应不同的策略。

9.根据权利要求7所述的一种容器网络隔离系统，其特征在于，还包括统一管理单元，统一管理单元通信连接每一生效的pod，用于统一管理所有生效了规则的pod。

10.一种计算机存储介质，其特征在于，其存储有计算机程序，所述计算机程序被处理器调用实现权利要求1-6任一所述的一种容器网络隔离方法。

一种容器网络隔离方法和系统

技术领域

[0001] 本方案涉及数据安全技术领域,尤其涉及一种容器网络隔离方法和系统。

背景技术

[0002] 容器网络隔离,一般使用k8s的Network Policy,实现对目标pod的入栈流量和出栈流量做IP网段、命名空间以及应用(Pod)做端口级的网络访问控制策略。

[0003] K8s的Network Policy虽然具备较强的可定制性,但是需要和label配置起来使用,配置不人性化,可读性差,维护困难。同时,仅定义一个Network Policy无法完成实际的网络隔离,还需要一个策略控制器(Policy Controller)进行策略的实现。策略控制器须由第三方网络组件提供,如Calico、Cilium、Weave-net等开源项目均支持Network Policy的实现。

发明内容

[0004] 本发明为了解决上述至少一项缺点,提供了一种容器网络隔离方法。

[0005] 一种容器网络隔离方法,包括以下步骤:

[0006] 创建k8s下的自定义 Network Policy资源,填写包括生效的k8s kind、namespace和name,以实现对对应的pod生效;

[0007] 在用户态程序中,监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中;

[0008] 所述iptables nfqueue规则添加在流量出栈后的第一个位置,用以拦截从容器内的应用程序发出的数据包;

[0009] 在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue。

[0010] 其中,所述监控Network Policy资源变化,包括以下步骤:

[0011] 监控 Network Policy 的变化,当Network Policy发生变化时,获取更新的Network Policy资源,将当前接收的Network Policy资源转换成作用的源地址IP、目的地址IP,以及目的地址端口,写入本地缓存。

[0012] 其中,所述监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中的方法包括:

[0013] 获取Network Policy中填写的需要生效的pod容器的pid,计算获得network namespace fd,

[0014] 根据network namespace fd将iptables nfqueue规则下发到对应pod容器的network namespace中。

[0015] 作为一种优选方案,一种容器网络隔离方法还包括:通过epoll统一管理所有pod,将所有生效了规则的pod的network namespace fd加入epoll中,

[0016] epoll监控到事件后,读取pod中出栈数据包。

[0017] 作为一种优选方案,所述本地缓存为hash多维缓存,

[0018] 通过所述hash多维缓存储包括:源地址IP、目的地址IP,以及携带优先级的策略列表,与拦截的数据包提取的源地址IP、目的地址IP,以及目的地址端口具有映射关系,不同目的地址端口对应不同的策略。

[0019] 作为一种优选方案,在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue的方法包括:

[0020] 响应于流量出栈后的第一个位置发生新事件,拦截数据包,并分析所述数据包携带的包括源地址IP、目的地址IP,以及目的地址端口信息,映射到hash多维缓存后,找到对应的策略,判断如何处置。

[0021] 本发明为了解决上述至少一项缺点,提供一种容器网络隔离系统,包括以下结构:

[0022] 网络策略单元,用于k8s下的Network Policy资源,填写生效的k8s kind、namespace和name,以实现对对应的pod生效;

[0023] 策略监控单元,用于在用户态程序中,监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中;

[0024] 所述iptables nfqueue规则添加在流量出栈后的第一个位置,用以拦截从容器内的应用程序发出的数据包;

[0025] 事件监控处置单元,用于在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue。

[0026] 作为一种优选方案,容器网络隔离系统还包括hash多维缓存单元,通讯连接用户态程序;

[0027] 所述hash多维缓存单元包括一维存储模块,二维存储模块和三维存储模块,

[0028] 一维存储模块的hash key包括源地址IP,二维存储模块的hash key包括目的地址IP,三维存储模块包括携带优先级的策略列表,不同目的地址端口对应不同的策略。

[0029] 作为一种优选方案,容器网络隔离系统还包括统一管理单元,统一管理单元通信连接每一生效的pod,用于统一管理所有生效了规则的pod。

[0030] 有益效果:(1)本方案使用iptables中的nfqueue,将内核态的握手包流量导入用户态中,根据重新定义的Network Policy自定义资源,实现对数据包的处置。灵活实现对K8s中的资源做网络隔离,降低运维成本和容器网络隔离资源的管理成本,且能够兼容所有主流的容器网络cni插件。

[0031] (2)本方案具有优秀的兼容性。本方案对出栈流量做控制,常见入栈流量中间经过cni和网络代理后,数据包发生变化,而出栈流量能够获取到容器网络的原始数据包,iptables nfqueue规则添加在流量出栈后的第一个位置,不会受到任何其他规则的影响,即使是容器网络插件cni在里面加的规则,所以该方案具备非常好的兼容性,在实践过程中适配目前已知的所有cni。

[0032] (3)创建Network Policy资源,填写生效的k8s kind、namespace和name,以实现对对应的pod生效。无需关注label配置,无需填写label,让规则可读性更强。

附图说明

[0033] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现

有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动性的前提下,还可以根据这些附图获得其他的附图。

[0034] 图1是容器网络隔离方法的整体流程图;

[0035] 图2是hash多维缓存单元示意图。

具体实施方式

[0036] 下面结合实施例对本发明做进一步的详细说明,以下实施例是对本发明的解释而本发明并不局限于以下实施例。

[0037] 名称解释:iptables,netflter/iptables(简称为iptables)组成Linux平台下的包过防火墙。iptables内置4个表,即flte表、nat表、mangle表和raw表,分别用于实现包过滤,网络地址转换、包重构(修改)和数据跟踪处理。

[0038] 实施例1:一种容器网络隔离系统,包括以下结构:

[0039] 网络策略单元,用于k8s下的自定义Network Policy资源,填写生效的k8s kind、namespace和name,以实现对对应的pod生效;

[0040] 策略监控单元,用于在用户态程序中,监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中;

[0041] 所述iptables nfqueue规则添加在流量出栈后的第一个位置,用以拦截从容器内的应用程序发出的数据包;

[0042] 事件监控处置单元,用于在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue。

[0043] 在k8s的架构上,master节点拥有Api-server,能够接收新定义的Network Policy的yaml,存入Api-server中。即创建新的Network Policy CRD。

[0044] 内核和用户态程序使用nfnetlink协议通信,当一个数据包入队列,内核向socket发送一个nfnetlink格式的消息,消息包含数据包数据和相关信息,用户态程序读取该socket就可以获取消息。

[0045] 在每个节点上分布用于监听Api-server的Daemonset Agent程序,用于监听Api-server中Network Policy的变化,生效新的Network Policy的规则。

[0046] 其中,策略监控单元执行于用户态程序下,使用k8s Daemonset控制器,让每个k8s node节点都能够启动一个用户态程序,用于用户态的处理。用户态程序监控集群的Network Policy资源变化。借助于k8s informer机制,list/watch Network Policy 资源的变化。

[0047] 作为一种优选的方案,配置定时监控单元,在最大同步时间内全量同步,在最小同步时间内增量同步。

[0048] 对于创建的新的Network Policy CRD 对应生效的pod,需要在pod network namespace 中添加iptables规则,规则如下:

[0049] iptables -t raw -I OUTPUT 1 -p tcp -syn -j NFQUEUE -queue-num=1 -queue-bypass;

[0050] 即,将该规则添加在nffilter raw表OUTPUT chain,该位置是容器网络进入nffilter后的第一个位置,在所有nffilter规则中处于优先级是最高的地位,也是

nffilter规则链在出栈时经过的第一个位置,相当于流量从容器内的应用程序发出后,就能够被nfqueue所拦截,不会受到任何其他规则的影响,即使是容器网络插件cni在里面加的规则,所以该方案具备非常好的兼容性。

[0051] 事件监控处置单元,用于监听nfnetlink socket,实现在用户态获取所有生效了规则的pod的出栈的数据包,分析该数据包,判断收放行,是否阻断,或者产生告警事件。

[0052] 用户态程序将拦截获取的数据包解析后,根据数据包中的payload字符数组,获得数据包的基本信息,该基本信息包括源地址IP、目的地址IP,以及目的地址端口。根据得到的数据包基本信息,遍历每个Network Policy CRD策略,判断端口是否在策略中,如果在策略中,生效对应的策略。

[0053] 作为一种优选的方案,容器网络隔离系统还包括统一管理单元,统一管理单元通信连接每一生效的pod,用于统一管理所有生效了规则的pod。

[0054] 启动epoll,将所有生效了规则的pod的network namespace fd加入epoll中,通过epoll统一管理所有pod,所有生效规则的pod流出的数据包均通过socket被事件监控处置单元截取。

[0055] 作为一种优选的方案,容器网络隔离系统还包括hash多维缓存单元,如图2所示,hash多维缓存单元,通讯连接用户态程序;所述hash多维缓存单元包括一维存储模块,二维存储模块和三维存储模块;一维存储模块的hash key包括源地址IP,二维存储模块的hash key包括目的地址IP,三维存储模块包括携带优先级的策略列表,不同目的地址端口对应不同的策略。

[0056] 在k8s中同一个集群中,负责实例转发的service和实例pod,都具备全局唯一的IP,在CRD中填写的是对应的controller名称,通过名称访问k8s-apiserver,获得对应资源的IP,当监测到更新的资源,经过程序处理后最终都会被转换成IP的形式,并保存在本地多维hash缓存中。

[0057] 实施例2:一种容器网络隔离方法,如图1所示,包括以下步骤:

[0058] 创建k8s下的自定义Network Policy资源,填写生效的k8s kind、namespace和name,以实现对对应的pod生效;

[0059] 在用户态程序中,监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中;

[0060] 所述iptables nfqueue规则添加在流量出栈后的第一个位置,用以拦截从容器内的应用程序发出的数据包;

[0061] 在用户态程序中,获取拦截的数据包,完成分析后将分析后的处置结果返回nfqueue。

[0062] 其中,所述监控Network Policy资源变化,包括以下步骤:

[0063] 监控 Network Policy 的变化,当Network Policy发生变化时,获取更新的Network Policy资源,将当前接收的Network Policy资源转换成作用的源地址IP、目的地址IP,以及目的地址端口,写入本地缓存。

[0064] 其中,所述监控Network Policy资源变化,下发iptables nfqueue规则到生效的pod 内核网络协议栈中的方法包括:

[0065] 获取Network Policy中填写的需要生效的pod容器的pid,计算获得network

namespace fd,根据network namespace fd将iptables nfqueue规则下发到对应pod容器的network namespace中。

[0066] 作为一种优选方案,所述本地缓存为hash多维缓存,通过所述hash多维缓存储包括:源地址IP、目的地址IP,以及携带优先级的策略列表,与拦截的数据包提取的源地址IP、目的地址IP,以及目的地址端口具有映射关系,不同目的地址端口对应不同的策略。

[0067] 作为一种优选方案,通过epoll统一管理所有pod,将所有生效了规则的pod的network namespace fd加入epoll中,

[0068] epoll监控到事件后,读取pod中出栈数据包。

[0069] 进一步,提供一种完整的容器网络隔离方法流程。

[0070] 创建一个新的network Policy CRD,在这个新的CRD 内部,填写生效的k8s kind、namespace和name,以此实现对controller下的pod生效。

[0071] 用户态程序启动,

[0072] 通过k8s informer监控Network Policy资源的更新;即调用k8s informer机制list/watch Network Policy 资源的变化;

[0073] 当Network Policy资源发生更新时,获取更新的Network Policy,将更新的Network Policy转换成作用的源地址IP、目的地址IP,以及目的地址端口,写入本地缓存;

[0074] 即,在用户态程序中,订阅Network Policy的变化,更新用户态缓存,缓存采用hash多维缓存,用于在用户态能够在O(1)的时间复杂度下,判断流量的处置结果,提高整体的运行效率,减少添加了规则后对pod流量QPS的影响。

[0075] 获取Network Policy中需要生效的容器的pid,pid的获取是通过运行时接口获得,通过 unix.Open("/proc/pid/ns/net") 获得network namespace fd;

[0076] 获得network namespace fd后,通过获取的该fd将规则下发到容器的network namespace中;同时将获取的fd加入到epoll中;

[0077] 当epoll监控到fd有新事件(有新的出栈流量)发生后,读取数据包,获取数据包的基本信息;数据包中包含了源地址IP、目的地址IP,以及目的地址端口,映射到hash多维缓存后,从hash多维缓存找到对应的策略,判断如何处置,处置结果包括阻断、放行,或产生告警事件;将处置结果通过socket发回nfqueue中,以实现处置结果。

[0078] 其中,用户态程序解析一个数据包时,需要组织一个nfnetlink格式的消息,消息中包含数据包在队列中的索引号,然后将消息发送给socket。考虑到用户态程序需要一个节点上所有生效了Network Policy的所有pod,所以使用epoll处理所有的socket。

[0079] 其中,本方案中,监控和拦截的数据包为握手流量包。

[0080] 所述集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时,可以存储在一个可读取存储介质中。基于这样的理解,本发明实施例的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来,该软件产品存储在一个存储介质中,包括若干指令用以使得一个设备(可以是单片机,芯片等)或处理器(processor)执行本发明各个实施例所述容器网络隔离方法的全部或部分步骤。而前述的存储介质包括:U盘、移动硬盘、只读存储器(Read-Only Memory,ROM)、随机存取存储器(Random Access Memory,RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

[0081] 一种电子设备，包括计算机存储介质和处理器，所述存储器用于存储一条或多条计算机指令，其中，所述一条或多条计算机指令被所述处理器执行以实现所述的容器网络隔离方法。

[0082] 在本申请所提供的几个实施例中，应该理解到，所揭露的装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如，所述单元或单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个装置，或一些特征可以忽略，或不执行。

[0083] 所述单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是一个物理单元或多个物理单元，即可以位于一个地方，或者也可以分布到多个不同地方。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0084] 另外，在本发明各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

[0085] 以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何在本发明揭露的技术范围内的变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应以所述权利要求的保护范围为准。

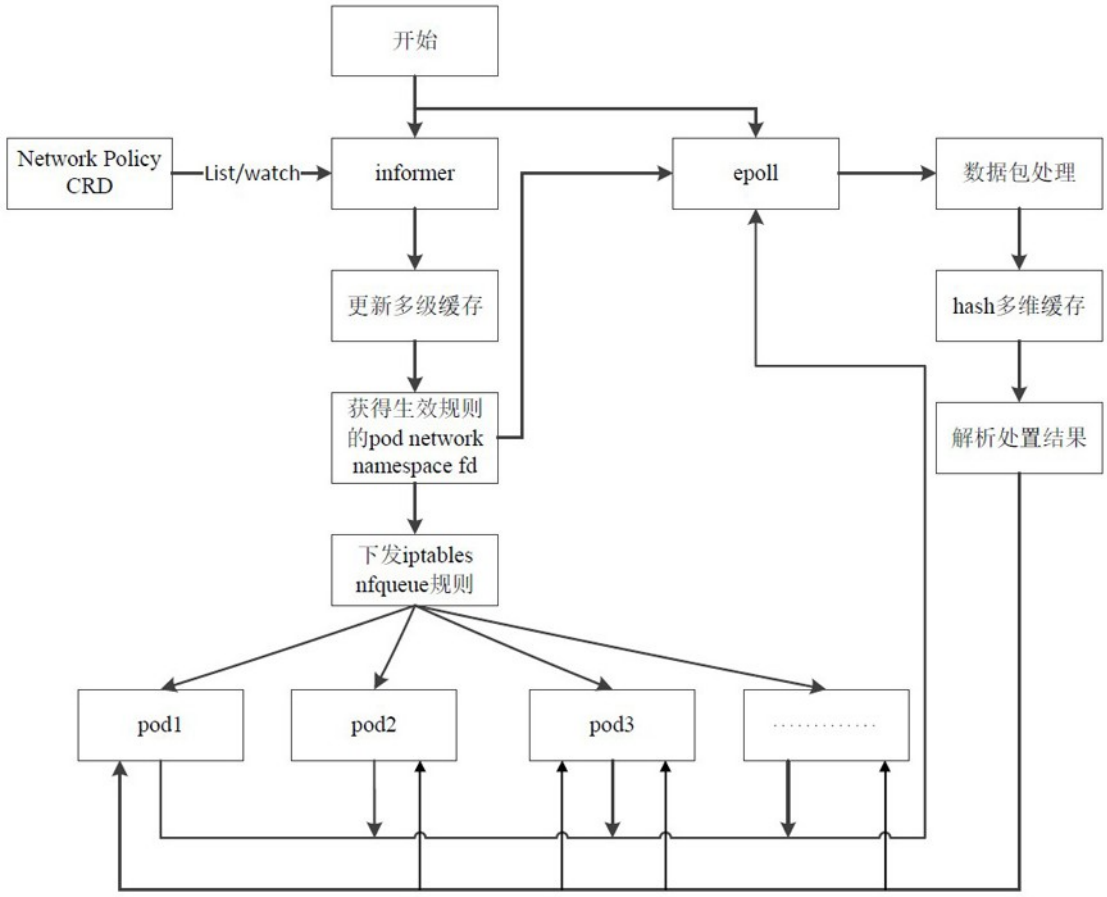

开始
Network Policy CRD
List/watch
informer
epoll
数据包处理
更新多级缓存
hash多维缓存
获得生效规则的pod network namespace fd
解析处置结果
下发iptables nfqueue规则
pod1
pod2
pod3
.....

图 1


源IP 1
源IP 2
源IP 3
哈希函数
目的IP 1
目的IP 1
目的IP 1
哈希函数
策略1
策略2
策略3
...
策略1
策略2
策略3
...
策略1
策略2
策略3
...

图 2