# NFTables: 设计动机、技术路线、试用与分析

## 设计动机

NFTables (Netfilter Tables) 是 Linux 内核中用于数据包过滤和分类的框架，旨在替代传统的 iptables/ip6tables/arptables/ebtables 工具集。其设计动机主要包括：

1. **解决iptables的扩展性问题**：iptables在大规模规则集时性能下降明显
2. **简化代码架构**：iptables代码库复杂且难以维护
3. **提供更灵活的表达能力**：支持更复杂的数据包匹配和处理逻辑
4. **统一各种表工具**：用一个框架替代多个分散的工具(iptables, ip6tables等)
5. **支持现代网络需求**：如更好的IPv6支持、更高效的规则处理

## 技术路线

NFTables采用了以下关键技术路线：

1. **基于Netfilter框架**：重用Linux内核的Netfilter钩子点
2. **新的虚拟机设计**：使用基于寄存器的虚拟机处理规则
3. **统一语法**：单一语法处理IPv4/IPv6/ARP等协议
4. **规则集优化**：支持规则集原子替换和增量更新
5. **JSON支持**：可通过JSON格式导入导出规则

## 试用与示例

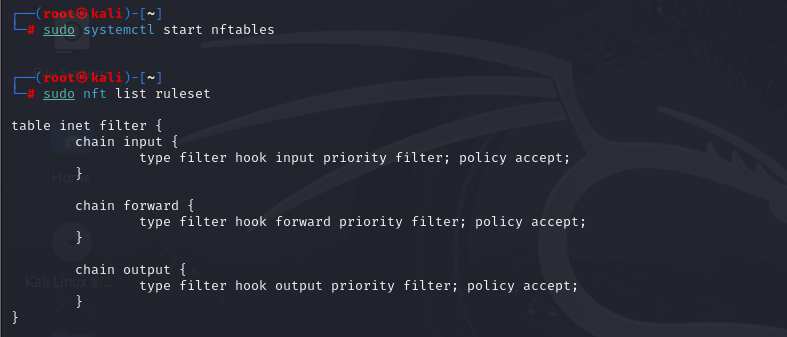
### 基本安装

在大多数Linux发行版上安装nftables：

sudo apt install nftables # Debian/Ubuntu

sudo yum install nftables # RHEL/CentOS

这里我直接使用了kali虚拟机，无需安装直接启动，下面是启动并查看规则集：



### 简单防火墙配置示例

# 清空现有规则

sudo nft flush ruleset



# 创建名为"filter"的表，处理IPv4流量

sudo nft add table ip filter



# 在filter表中创建input链

sudo nft add chain ip filter input \{ type filter hook input priority 0 \; \}

# 允许已建立的连接

sudo nft add rule ip filter input ct state established,related accept



# 允许SSH(22端口)和HTTP(80端口)

sudo nft add rule ip filter input tcp dport \{22, 80\} accept



# 允许ICMP(ping)

sudo nft add rule ip filter input icmp type echo-request accept



# 默认拒绝策略

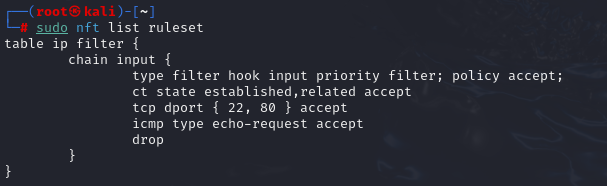
sudo nft add rule ip filter input drop



# 查看当前规则集

sudo nft list ruleset

下面就是经过我设置的一个个性化的防火墙规则集



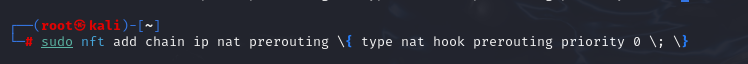
### NAT配置示例

# 创建nat表

sudo nft add table ip nat



# 创建prerouting和postrouting链

sudo nft add chain ip nat prerouting \{ type nat hook prerouting priority 0 \; \}

sudo nft add chain ip nat postrouting \{ type nat hook postrouting priority 100 \; \}



# 启用IP伪装(MASQUERADE)

sudo nft add rule ip nat postrouting oifname "eth0" masquerade



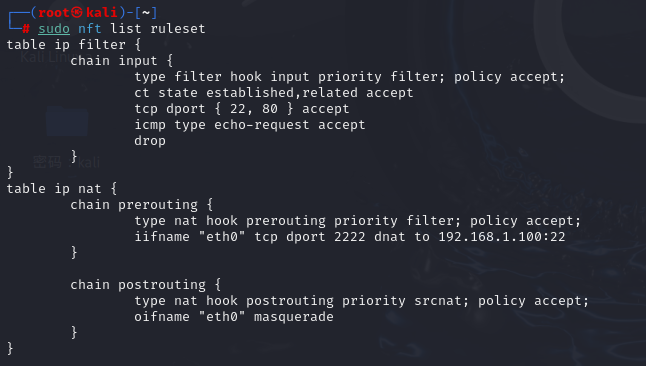
# 端口转发(将外部2222端口转发到内部22端口)

sudo nft add rule ip nat prerouting iifname "eth0" tcp dport 2222 dnat to 192.168.1.100:22



# 查看当前规则集

sudo nft list ruleset



**上述规则集的效果：**

**1. 防火墙规则（filter 表）**

**允许已建立的连接：**放行属于现有网络会话的流量（如已经建立的 SSH 或网页浏览连接），防止意外中断。

**开放指定端口：**明确允许访问 SSH（22） 和 HTTP（80），确保这些服务可被外部访问。

**允许 ICMP（ping）**：使本机能响应 ping 请求，便于网络诊断。

**默认拒绝所有其他流量：**未明确允许的访问一律阻断，提升安全性（比如阻止随机端口扫描或恶意连接）。

实际影响：

外部设备可以正常访问本机的 SSH 和 Web 服务，但无法连接其他未开放的端口（如 MySQL 的 3306）。

本机可以主动向外发起连接（如浏览网页），但外部未经允许的入站连接会被丢弃。

**2. NAT 规则（nat 表）**

**IP 伪装（MASQUERADE）：**让内网设备通过本机的公网 IP 上网（如手机/虚拟机共享 Kali 的网络）。

内网设备发出的流量会被“伪装”成来自 Kali 的公网 IP（58.194.169.228）。

**端口转发（DNAT）**：将访问 Kali 公网 IP 2222 端口 的流量转发到内网某台机器的 22 端口（SSH）。

外部用户连接 kali\_ip:2222 时，实际会访问内网的 192.168.1.100:22。

实际影响：

内网设备无需公网 IP 即可上网（如虚拟机通过 Kali 访问互联网）。

可以通过 Kali 的公网 IP 间接访问内网服务（如远程管理内网服务器的 SSH）。

[更多功能](https://blog.csdn.net/i89211/article/details/146089016?ops_request_misc=&request_id=&biz_id=102&utm_term=nftables&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-3-146089016.142%5ev102%5epc_search_result_base7&spm=1018.2226.3001.4187)

[nftables Wiki](https://wiki.nftables.org/wiki-nftables/index.php/Main_Page)

## 原理与机制分析

### 1. 规则处理虚拟机

NFTables的核心是一个基于寄存器的虚拟机，它负责解释和执行规则。与iptables的线性规则处理不同，NFTables的虚拟机可以：

* 更高效地处理复杂规则
* 支持条件跳转和更灵活的控制流
* 减少内核和用户空间之间的上下文切换

### 2. 表、链和规则结构

* **表(Table)**：最高层次的容器，按协议族(ip, ip6, inet等)划分
* **链(Chain)**：包含规则列表，可以是基链(有钩子点)或常规链
* **规则(Rule)**：由匹配表达式和动作语句组成

### 3. 规则集原子性

NFTables支持原子规则集更新，整个规则集可以作为一个单元替换，避免了iptables中规则逐条添加删除时可能出现的不一致状态。

### 4. 集合和字典

NFTables引入了集合和字典的概念，允许高效地匹配多个值：

# 定义端口集合

nft add set ip filter webports { type inet\_service \; flags interval \; elements = { 80, 443, 8080 } }

# 使用集合的规则

nft add rule ip filter input tcp dport @webports accept

### 5. 性能优势

NFTables相比iptables的主要性能优势来自：

1. **更少的规则评估**：通过集合和字典减少线性匹配需求
2. **更少的内核/用户空间切换**：批量规则加载
3. **更高效的字节码**：基于寄存器的虚拟机比iptables的基于栈的评估更高效

### 6. 向后兼容性

NFTables提供了iptables的兼容层(xtables-nft)，允许在过渡期间继续使用iptables语法。

## nftables和iptables的主要区别

**语法不同**

iptables 使用基于 getopt\_long() 的解析器，其中关键字总是以双破折号（--）开头，例如 --key，或者单个破折号（-）后跟一个字母，例如 -p tcp。

nftables 则使用了一种受 tcpdump 启发的紧凑语法。这种语法更加直观和灵活，让用户能够更轻松地编写和理解规则。

**表和链的完全可配置性**

iptables 预设了多个表和基础链，即使你只需要其中的一个，所有的表和链都会被注册。这可能导致性能问题，即使未使用的基础链也可能对性能产生负面影响。

nftables 没有预设的表和链。你需要显式地定义每个表，并且只向其中添加你明确需要的对象（链、集合、映射、流表和状态对象）。这样，你就可以只注册你需要的基础链，并根据你的特定数据包处理管道选择表和链的名称以及 netfilter 钩子优先级。

**单个规则可以执行多个操作**

iptables 的规则由匹配条件和单个目标动作组成。一旦匹配条件满足，就会执行目标动作（如接受、拒绝或跳转到另一个链）。

nftables 的规则由零个或多个表达式后跟一个或多个语句组成。每个表达式测试数据包是否与特定的负载字段或数据包/流元数据匹配。多个表达式从左到右线性评估：如果第一个表达式匹配，则评估下一个表达式，依此类推。如果所有表达式都匹配，则执行该规则的语句。每个语句执行一个操作，如设置 netfilter 标记、计数数据包、记录数据包或做出裁决（如接受、丢弃数据包或跳转到另一个链）。与表达式一样，多个语句也是从左到右线性评估的：单个规则可以通过使用多个语句来执行多个操作。但请注意，裁决语句会自然结束规则的执行。

**没有内置的每链和每规则计数器**

在 iptables 中，计数器是内置的，用于跟踪每个链和规则的匹配次数。但在 nftables 中，计数器是可选的，可以根据需要启用它们。这种灵活性允许用户只在需要时跟踪特定规则或链的性能指标，从而节省系统资源。

**更好的动态规则集更新支持**

iptables 使用一个整体的数据块（monolithic blob）来存储其规则集，这意呀着在添加或删除规则时，可能需要重新加载整个规则集，这可能会影响系统的性能。而 nftables 的规则集在内部以链表的形式表示，这使得添加或删除规则时只影响相关的部分，而不会影响整个规则集的其余部分，从而简化了内部状态信息的维护。

**简化的双栈 IPv4/IPv6 管理**

nftables 的 inet 家族允许注册能够同时查看 IPv4 和 IPv6 流量的基础链。这意味着不再需要依赖脚本来复制你的规则集以同时支持 IPv4 和 IPv6，从而简化了配置和管理过程。

**新的通用集合基础设施**

nftables 引入了与核心紧密集成的新的通用集合基础设施，它允许高级配置，如映射（maps）、决策映射（verdict maps）和间隔（intervals），以实现面向性能的包分类。最重要的是，可以使用任何受支持的选择器来对流量进行分类，这为复杂的网络场景提供了更多的灵活性和控制力。

**支持拼接**

自 Linux 内核 4.1 版本以来，nftables 支持将多个关键字拼接起来，并将它们与映射和裁决映射组合使用。

**无需内核升级即可支持新协议**

在传统的防火墙配置中，支持新协议通常需要升级内核。而 nftables 采用了新的虚拟机方法，使得支持新协议通常不需要新的内核版本，而只需要相对简单的 nft 用户空间软件更新。这降低了维护成本，并提高了系统的灵活性和可扩展性。