Guia Completo de nftables para Iniciantes

Sumário

Introdução ao nftables

O que é um firewall?

Imagine um firewall como um porteiro de um prédio. Ele decide quem pode entrar, quem pode sair e quais andares cada pessoa pode acessar. No mundo dos computadores, o firewall controla o tráfego de dados que entra e sai do seu sistema, permitindo ou bloqueando conexões baseadas em regras que você define.

nftables vs iptables: Uma Breve História

Por muitos anos, o **iptables** foi a ferramenta padrão para configurar firewalls no Linux. O **nftables** é sua evolução moderna, introduzido para resolver limitações do iptables. Pense no iptables como um telefone fixo antigo e no nftables como um smartphone: ambos fazem ligações, mas o smartphone é mais eficiente, flexível e fácil de usar.

Principais vantagens do nftables:

- Sintaxe mais clara e consistente
- Melhor performance (processa regras mais rapidamente)

- Configuração unificada (um único comando para IPv4 e IPv6)
- Menos código duplicado

Nossa Topologia de Rede

Neste guia, trabalharemos com a seguinte estrutura:

```
INTERNET (WAN)
       |
   [enp1s0] ← Interface WAN (externa)
       |
  [FIREWALL]
    /     \
[enp2s0] [enp3s0] ← Interfaces LAN (internas)
   |        |
 LAN1     LAN2
```

Entenda as interfaces:

- **enp1s0 (WAN)**: conectada à internet, recebe tráfego externo
- **enp2s0 (LAN1)**: rede interna 1, por exemplo: 192.168.1.0/24
- **enp3s0 (LAN2)**: rede interna 2, por exemplo: 192.168.2.0/24

Conceitos Fundamentais

1. Tables (Tabelas) e Famílias (Families)

Tabelas são como departamentos em uma empresa. Cada departamento tem uma função específica. Cada tabela deve ter uma família de endereços definida. A família de endereços de uma tabela define que tipos de endereços a tabela processa. Você pode definir uma das seguintes famílias de endereços ao criar uma tabela:

- **ip**: processa tráfego IPv4
- **ip6**: processa tráfego IPv6
- **inet**: processa IPv4 e IPv6 simultaneamente (mais prático!)

- **arp**: processa tráfego ARP
- **bridge**: processa tráfego em bridges de rede
- **netdev**: Combina pacotes de entrada

Recomendação para iniciantes: Use tabelas do tipo **inet** para simplificar sua configuração.

2. Chains (Correntes)

Chains são como pontos de verificação onde o tráfego passa. Imagine uma esteira de aeroporto com vários checkpoints de segurança. As chains principais são:

- **input**: tráfego que ENTRA no firewall (destino é o próprio firewall)
- **output**: tráfego que SAI do firewall (origem é o próprio firewall)
- **forward**: tráfego que PASSA PELO firewall (de uma interface para outra)
- **prerouting**: processa pacotes ANTES do roteamento (usado para DNAT)
- **postrouting**: processa pacotes DEPOIS do roteamento (usado para SNAT/Masquerade)

3. Rules (Regras)

Regras são as instruções específicas: "se o pacote vem da internet, bloqueie" ou "se é para a porta 80, permita". Cada regra tem:

- **Conditions**: condições que o pacote deve atender (porta, IP de origem, protocolo)
- **Actions**: o que fazer com o pacote (accept, drop, reject)

4. Policy (Política Padrão)

A policy é o comportamento padrão quando nenhuma regra específica se aplica. É como uma placa "proibido estacionar, exceto moradores". Existem duas abordagens:

- **Política restritiva (recomendada)**: bloqueie tudo por padrão, permita apenas o necessário
- **Política permissiva**: permita tudo por padrão, bloqueie o que for perigoso

5. Sets (Conjuntos)

Sets são listas de valores que você pode reutilizar. Em vez de criar 10 regras para 10 IPs diferentes, você cria um set com os 10 IPs e usa uma única regra. Pense em uma lista de convidados VIP em uma festa.

Estrutura do Arquivo nftables.conf

O arquivo de configuração principal do nftables fica em `/etc/nftables.conf` no Debian. Este arquivo é carregado automaticamente na inicialização do sistema.

Estrutura Básica

```
#!/usr/sbin/nft -f
# Limpa todas as configurações existentes
flush ruleset

# Define a tabela principal
table inet filter {
    # Define chains
    chain input {
        # Regras para tráfego entrando no firewall
    }

    chain forward {
        # Regras para tráfego passando pelo firewall
    }

    chain output {
        # Regras para tráfego saindo do firewall
    }
}
```

Exemplo Completo e Comentado para Nossa Topologia

```
#!/usr/sbin/nft -f
# ============================================
# Configuração de Firewall nftables
# Topologia: 1 WAN (enp1s0) + 2 LANs (enp2s0, enp3s0)
# ============================================

# Limpa todas as regras anteriores
flush ruleset

# ============================================
# TABELA DE FILTRAGEM
# ============================================
table inet filter {

    # Define um conjunto de IPs confiáveis (exemplo)
    set trusted_ips {
        type ipv4_addr
        elements = { 192.168.1.10, 192.168.2.50 }
    }

    # ----------------------------------------
    # CHAIN INPUT - Tráfego para o firewall
    # ----------------------------------------
    chain input {
        type filter hook input priority 0; policy drop;

        # Aceita tráfego da interface loopback (comunicação interna do sistema)
        iif "lo" accept comment "Aceita loopback"

        # Aceita conexões já estabelecidas ou relacionadas
        ct state established,related accept comment "Aceita conexões estabelecidas"

        # Aceita ICMP (ping) das LANs
        iifname { "enp2s0", "enp3s0" } icmp type echo-request accept comment "Aceita pi

        # Aceita SSH apenas das LANs
```

```
        iifname { "enp2s0", "enp3s0" } tcp dport 22 accept comment "SSH das LANs"

        # Log de pacotes descartados (útil para debugging)
        log prefix "INPUT DROP: " drop
    }

    # ---------------------------------------
    # CHAIN FORWARD - Tráfego passando pelo firewall
    # ---------------------------------------
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Aceita conexões já estabelecidas
        ct state established,related accept comment "Aceita conexões estabelecidas"

        # Permite LAN1 e LAN2 acessarem a internet (WAN)
        iifname { "enp2s0", "enp3s0" } oifname "enp1s0" accept comment "LANs para inter

        # Permite comunicação entre LAN1 e LAN2
        iifname "enp2s0" oifname "enp3s0" accept comment "LAN1 para LAN2"
        iifname "enp3s0" oifname "enp2s0" accept comment "LAN2 para LAN1"

        # Bloqueia tráfego da WAN para as LANs (a menos que seja estabelecido)
        # Esta regra é implícita pela policy drop, mas pode ser explicitada
        log prefix "FORWARD DROP: " drop
    }

    # ---------------------------------------
    # CHAIN OUTPUT - Tráfego saindo do firewall
    # ---------------------------------------
    chain output {
        type filter hook output priority 0; policy accept;
        # Normalmente permitimos tudo que sai do firewall
    }
}

# ========================================
# TABELA DE NAT
```

```
# ========================================
table inet nat {

    # ----------------------------------------
    # CHAIN PREROUTING - Antes do roteamento
    # Usado para DNAT (port forwarding)
    # ----------------------------------------
    chain prerouting {
        type nat hook prerouting priority -100;

        # Exemplo: Redireciona porta 8080 da WAN para servidor web em LAN1
        iifname "enp1s0" tcp dport 8080 dnat to 192.168.1.100:80 comment "Port forward
    }

    # ----------------------------------------
    # CHAIN POSTROUTING - Depois do roteamento
    # Usado para SNAT/Masquerade
    # ----------------------------------------
    chain postrouting {
        type nat hook postrouting priority 100;

        # Masquerade: traduz IPs internos para o IP da WAN
        oifname "enp1s0" masquerade comment "NAT para internet"
    }
}
```

Explicação Detalhada das Seções

1. Flush ruleset: limpa todas as regras existentes para começar do zero. Cuidado: se você executar isso remotamente sem ter regras que permitam SSH, perderá a conexão!

2. Policy drop vs accept:

- `policy drop` significa "bloquear tudo, exceto o que eu permitir explicitamente" (mais seguro)

- `policy accept` significa "permitir tudo, exceto o que eu bloquear explicitamente" (menos seguro)

3. ct state (connection tracking): o firewall "lembra" de conexões ativas. Se você iniciou uma conexão da LAN para a internet, as respostas são automaticamente aceitas.

4. iifname e oifname:

- `iifname` = interface de entrada (input interface)
- `oifname` = interface de saída (output interface)

5. Priority: número que define a ordem de processamento. Números menores são processados primeiro. Para NAT, prerouting usa prioridade negativa para ser processado antes da filtragem.

Comandos Essenciais

1. Visualizando Configurações

Listar todas as regras:

```
sudo nft list ruleset
```

Resultado esperado:

```
table inet filter {
    chain input {
        type filter hook input priority 0; policy drop;
        iif "lo" accept
        ...
    }
}
```

Listar apenas uma tabela específica:

```
sudo nft list table inet filter
```

Listar apenas uma chain específica:

```
sudo nft list chain inet filter input
```

2. Adicionando Regras Dinamicamente

Sintaxe geral:

```
sudo nft add rule [families] [tables] [chain] [conditions] [action]
```

Exemplo 1: Permitir HTTP na chain forward

```
sudo nft add rule inet filter forward iifname "enp2s0" tcp dport 80 accept
```

Explicação: Esta regra permite que dispositivos na LAN1 (enp2s0) acessem servidores web (porta 80).

Exemplo 2: Bloquear um IP específico

```
sudo nft add rule inet filter input ip saddr 203.0.113.5 drop
```

Explicação: Bloqueia qualquer tráfego vindo do IP 203.0.113.5.

Exemplo 3: Adicionar regra no início da chain (posição 0)

```
sudo nft insert rule inet filter input position 0 tcp dport 443 accept
```

Diferença entre add e insert:

- `add` : adiciona a regra no **final** da chain
- `insert` : adiciona a regra no **início** da chain (posição 0) ou em posição específica

3. Deletando Regras

Método 1: Por handle (identificador único)

Primeiro, liste as regras com handles:

```
sudo nft -a list chain inet filter input
```

Resultado:

```
chain input {
    type filter hook input priority 0; policy drop;
    iif "lo" accept # handle 5
    tcp dport 22 accept # handle 8
    tcp dport 80 accept # handle 12
}
```

Agora delete a regra específica:

```
sudo nft delete rule inet filter input handle 12
```

Método 2: Deletar chain inteira

```
sudo nft delete chain inet filter input
```

⚠️ **AVISO:** Deletar uma chain remove todas as suas regras!

4. Limpando Configurações

Limpar todas as regras de uma tabela:

```
sudo nft flush table inet filter
```

Limpar todas as regras de uma chain:

```
sudo nft flush chain inet filter input
```

Limpar TUDO (todas as tabelas e regras):

```
sudo nft flush ruleset
```

⚠️ **PERIGO:** Este comando remove todas as regras de firewall! Use com extrema cautela, especialmente em conexões remotas.

5. Salvando e Restaurando Configurações

Salvar configuração atual em arquivo:

```
sudo nft list ruleset > /etc/nftables.conf
```

Carregar configuração de arquivo:

```
sudo nft -f /etc/nftables.conf
```

Recarregar o serviço nftables:

```
sudo systemctl reload nftables
```

Verificar status do serviço:

```
sudo systemctl status nftables
```

6. Trabalhando com Sets

Criar um set:

```
sudo nft add set inet filter blacklist { type ipv4_addr \; }
```

Adicionar elementos ao set:

```
sudo nft add element inet filter blacklist { 198.51.100.5, 198.51.100.6 }
```

Usar o set em uma regra:

```
sudo nft add rule inet filter input ip saddr @blacklist drop
```

Listar elementos do set:

```
sudo nft list set inet filter blacklist
```

Regras Básicas: NAT, Filtragem e Port Forwarding

1. NAT (Network Address Translation)

NAT permite que múltiplos dispositivos em uma rede interna compartilhem um único IP público.

Conceito: Imagine um prédio com muitos apartamentos, mas apenas um endereço postal. O porteiro (NAT) recebe toda a correspondência e distribui para os apartamentos corretos.

Masquerade (SNAT Dinâmico)

Usado quando seu IP externo é dinâmico (comum em conexões domésticas).

```
table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp1s0" masquerade
    }
}
```

O que acontece:

1. Computador na LAN1 (192.168.1.50) quer acessar google.com
2. Pacote sai com IP origem 192.168.1.50
3. Firewall substitui por seu IP público (ex: 203.0.113.10)
4. Google responde para 203.0.113.10
5. Firewall traduz de volta para 192.168.1.50

SNAT Estático

Usado quando você tem um IP público fixo.

```
table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp1s0" ip saddr 192.168.1.0/24 snat to 203.0.113.10
    }
}
```

2. Port Forwarding (DNAT)

Permite que serviços internos sejam acessíveis da internet.

Cenário: Você tem um servidor web na LAN1 (192.168.1.100) e quer que ele seja acessível da internet.

Exemplo 1: Redirecionar porta 80 (HTTP)

```
table inet nat {
    chain prerouting {
        type nat hook prerouting priority -100;
        iifname "enp1s0" tcp dport 80 dnat to 192.168.1.100
    }
}
```

Também é necessário permitir na chain forward:

```
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;
        iifname "enp1s0" oifname "enp2s0" tcp dport 80 accept
        ct state established,related accept
    }
}
```

Exemplo 2: Redirecionar porta diferente

Redirecionar porta 8080 externa para porta 80 interna:

```
table inet nat {
    chain prerouting {
        type nat hook prerouting priority -100;
        iifname "enp1s0" tcp dport 8080 dnat to 192.168.1.100:80
    }
}
```

3. Filtragem Básica

Exemplo 1: Bloquear acesso a site específico

```
# Bloquear acesso ao IP do site (exemplo)
sudo nft add rule inet filter forward ip daddr 93.184.216.34 drop
```

Exemplo 2: Limitar taxa de conexões (proteção contra flood)

```
sudo nft add rule inet filter input tcp dport 22 limit rate 5/minute accept
```

Explicação: Aceita apenas 5 conexões SSH por minuto, protegendo contra ataques de força bruta.

Exemplo 3: Permitir apenas IPs específicos acessarem SSH

```
table inet filter {
    set admin_ips {
        type ipv4_addr
        elements = { 192.168.1.10, 192.168.2.20 }
    }

    chain input {
        type filter hook input priority 0; policy drop;
        tcp dport 22 ip saddr @admin_ips accept
    }
}
```

4. Regras Entre LANs

Permitir apenas LAN1 acessar LAN2 (unidirecional)

```
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # LAN1 pode acessar LAN2
        iifname "enp2s0" oifname "enp3s0" accept

        # LAN2 NÃO pode iniciar conexões para LAN1
        # (mas pode responder a conexões iniciadas por LAN1)
        ct state established,related accept
    }
}
```

Bloquear comunicação entre LANs

```
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Bloqueia tráfego entre LANs
        iifname "enp2s0" oifname "enp3s0" drop
        iifname "enp3s0" oifname "enp2s0" drop

        # Permite LANs acessarem internet
        iifname { "enp2s0", "enp3s0" } oifname "enp1s0" accept
        ct state established,related accept
    }
}
```

Cenários Práticos

Cenário 1: Configuração Básica de Gateway

Situação: Você quer que as duas LANs tenham acesso à internet, mas que o firewall seja protegido.

```
#!/usr/sbin/nft -f
flush ruleset

table inet filter {
    chain input {
        type filter hook input priority 0; policy drop;
        iif "lo" accept
        ct state established,related accept
        iifname { "enp2s0", "enp3s0" } tcp dport 22 accept
        iifname { "enp2s0", "enp3s0" } icmp type echo-request accept
    }

    chain forward {
        type filter hook forward priority 0; policy drop;
        ct state established,related accept
        iifname { "enp2s0", "enp3s0" } oifname "enp1s0" accept
    }

    chain output {
        type filter hook output priority 0; policy accept;
    }
}

table inet nat {
    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp1s0" masquerade
    }
}
```

Não esqueça de habilitar IP forwarding:

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -w net.ipv6.conf.all.forwarding=1

# Tornar permanente
echo "net.ipv4.ip_forward=1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv6.conf.all.forwarding=1" | sudo tee -a /etc/sysctl.conf
```

Cenário 2: Servidor Web na LAN1 Acessível da Internet

Situação: Servidor web em 192.168.1.100 deve ser acessível na porta 80 da internet.

```
table inet nat {
    chain prerouting {
        type nat hook prerouting priority -100;
        iifname "enp1s0" tcp dport 80 dnat to 192.168.1.100
    }

    chain postrouting {
        type nat hook postrouting priority 100;
        oifname "enp1s0" masquerade
    }
}

table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;

        # Permite conexões da internet para o servidor web
        iifname "enp1s0" oifname "enp2s0" ip daddr 192.168.1.100 tcp dport 80 accept

        # Permite respostas e tráfego das LANs
        ct state established,related accept
        iifname { "enp2s0", "enp3s0" } oifname "enp1s0" accept
    }
}
```

Cenário 3: LAN1 (Escritório) e LAN2 (Convidados) Isoladas

Situação: LAN1 tem acesso total, LAN2 só acessa internet e tem restrições.

```
table inet filter {
    chain input {
        type filter hook input priority 0; policy drop;
        iif "lo" accept
        ct state established,related accept

        # Apenas LAN1 pode acessar SSH do firewall
        iifname "enp2s0" tcp dport 22 accept

        # Ambas podem pingar o firewall
        iifname { "enp2s0", "enp3s0" } icmp type echo-request accept
    }

    chain forward {
        type filter hook forward priority 0; policy drop;
        ct state established,related accept

        # LAN1: acesso total
        iifname "enp2s0" oifname "enp1s0" accept
        iifname "enp2s0" oifname "enp3s0" accept

        # LAN2: apenas internet, sem acesso à LAN1
        iifname "enp3s0" oifname "enp1s0" accept

        # Bloqueia LAN2 tentando acessar LAN1
        iifname "enp3s0" oifname "enp2s0" drop
    }
}
```

Cenário 4: Múltiplos Port Forwards

Situação: Você tem vários servidores internos que precisam ser acessíveis.

```
table inet nat {
    chain prerouting {
        type nat hook prerouting priority -100;

        # Web server na LAN1
        iifname "enp1s0" tcp dport 80 dnat to 192.168.1.100:80
        iifname "enp1s0" tcp dport 443 dnat to 192.168.1.100:443

        # Servidor de email na LAN1
        iifname "enp1s0" tcp dport 25 dnat to 192.168.1.110:25
        iifname "enp1s0" tcp dport 587 dnat to 192.168.1.110:587

        # Servidor SSH na LAN2 (porta não-padrão)
        iifname "enp1s0" tcp dport 2222 dnat to 192.168.2.50:22
    }
}

table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;
        ct state established,related accept

        # Permite os forwards acima
        iifname "enp1s0" oifname "enp2s0" ip daddr 192.168.1.100 tcp dport { 80, 443 }
        iifname "enp1s0" oifname "enp2s0" ip daddr 192.168.1.110 tcp dport { 25, 587 }
        iifname "enp1s0" oifname "enp3s0" ip daddr 192.168.2.50 tcp dport 22 accept

        # LANs podem acessar internet
        iifname { "enp2s0", "enp3s0" } oifname "enp1s0" accept
    }
}
```

Erros Comuns e Como Evitá-los

---

export_on_save:
puppeteer: true # export PDF on save
puppeteer: ["pdf", "png"] # export PDF and PNG files on save
puppeteer: ["png"] # export PNG file on save

1. ❌ Perder acesso SSH ao configurar remotamente

Problema: Você aplica `policy drop` na chain input sem permitir SSH primeiro.

Solução: SEMPRE permita SSH antes de mudar a policy:

```
# CORRETO: Permitir SSH primeiro
sudo nft add rule inet filter input tcp dport 22 accept
sudo nft add chain inet filter input { type filter hook input priority 0 \; policy drop

# ERRADO: Mudar policy primeiro
sudo nft add chain inet filter input { type filter hook input priority 0 \; policy drop
# Agora você perdeu acesso!
```

Dica: Teste sempre localmente ou use `at` para reverter automaticamente:

```
sudo nft -f /etc/nftables.conf && sleep 60 && sudo nft flush ruleset
```

2. ❌ Esquecer de habilitar IP forwarding

Problema: Configurou NAT e forward rules, mas as LANs não acessam a internet.

Sintoma: `ping 8.8.8.8` não funciona das LANs, mas funciona do próprio firewall.

Solução:

```
# Verificar se está habilitado
cat /proc/sys/net/ipv4/ip_forward
# Se retornar 0, não está habilitado

# Habilitar temporariamente
sudo sysctl -w net.ipv4.ip_forward=1

# Habilitar permanentemente
echo "net.ipv4.ip_forward=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

3. ❌ Esquecer regras de estado (ct state)

Problema: Permitiu tráfego de saída, mas as respostas não voltam.

Exemplo errado:

```
chain forward {
    type filter hook forward priority 0; policy drop;
    iifname "enp2s0" oifname "enp1s0" accept
    # Faltou: ct state established,related accept
}
```

Solução: Sempre inclua:

```
ct state established,related accept
```

4. ❌ Ordem incorreta das regras

Problema: Regras mais específicas depois de regras genéricas são ignoradas.

Exemplo errado:

```
# Esta regra vem primeiro e bloqueia tudo
ip saddr 0.0.0.0/0 drop

# Esta nunca será processada
ip saddr 192.168.1.10 accept
```

Solução: Regras mais específicas devem vir ANTES:

```
# CORRETO
ip saddr 192.168.1.10 accept
ip saddr 0.0.0.0/0 drop
```

5. ❌ Confundir iifname com oifname

Problema: Usar a interface errada na regra.

Lembre-se:

- `iifname` = interface de **entrada** (input interface)
- `oifname` = interface de **saída** (output interface)

Exemplo para permitir LAN1 acessar internet:

```
# CORRETO
iifname "enp2s0" oifname "enp1s0" accept

# ERRADO
iifname "enp1s0" oifname "enp2s0" accept  # Isso é tráfego da internet para LAN!
```

6. ❌ Não salvar configurações

Problema: Configurou tudo via linha de comando, reiniciou o sistema e perdeu tudo.

Solução: Salve suas configurações:

```
sudo nft list ruleset > /etc/nftables.conf
sudo systemctl enable nftables
```

7. ❌ Sintaxe incorreta ao adicionar regras

Problema: Esquecer vírgulas, usar chaves incorretamente.

Errado:

```
iifname { "enp2s0" "enp3s0" } accept  # Falta vírgula
```

Correto:

```
iifname { "enp2s0", "enp3s0" } accept
```

8. ❌ Não testar após cada mudança

Problema: Adicionar 10 regras de uma vez e não saber qual causou o problema.

Solução: Teste incrementalmente:

```
# 1. Adicione uma regra
sudo nft add rule inet filter input tcp dport 80 accept

# 2. Teste
curl http://localhost

# 3. Se funcionar, adicione outra regra e teste novamente
```

Saiba mais:

GitHub: UC07-Instalacao_Config_Servidores_Linux

Netfilter: The netfilter.org "nftables" project

Wiki: nftables