



Linux Cluster in Theorie und Praxis

iptables und nftables

Alfred Krohmer

4. März 2014



- 1 Einführung
- 2 Rückblick / bisherige Firewall-Lösungen
- 3 Funktionsweise iptables vs. nftables
- 4 Syntax und Tools
- 5 Performance-Vergleich
- 6 Schlussfolgerung
- 7 Quellen

Zielstellungen bei der Entwicklung bei nftables

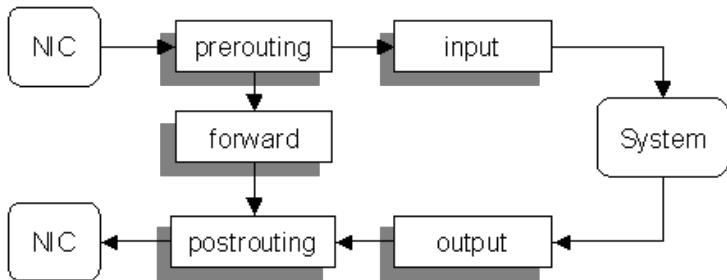
- Vereinfachung der Kernel-ABI
- Vermeidung von Code-Redundanz
- effizientere Abarbeitung der Regeln
- bessere Fehlermeldungen

- 1994: ipfw
- 1996: ipfwadm
- 1999: ipchains
- 2000: iptables
- **2014: nftables**

iptables:

- nur für IPv4
- andere Tools für andere Protokolle:
 - ip6tables
 - arptables
 - ebtables
- für jedes Protokoll eine eigenständige Implementierung im Kernel
- Code für jedes Protokoll sehr spezifisch
 - viel replizierter Code
 - hohe Performance

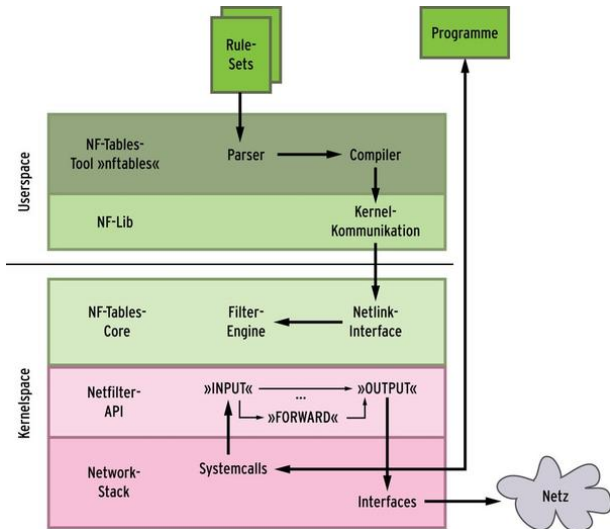
Funktionsweise iptables vs. nftables



nftables:

- ein Tool für alle Protokolle (IPv4, IPv6, Ethernet-Bridging, ARP)
- einheitliche Schnittstelle zum Kernel
- Implementierung als kleine virtuelle Maschine im Kernel
- Regeln werden im Userspace zu Byte-Code kompiliert
- Byte-Code kann auf Feldern und Bits der Pakete Operationen ausführen:
 - vergleichen (matching) → bedingte Sprünge
 - arithmetische und logische Operationen
 - beliebige Änderungen am Paketinhalt
- atomares Ersetzen von Regeln über Netlink-Transaktionen
- funktioniert mit bisher verfügbaren Tools noch nicht effizient

Funktionsweise iptables vs. nftables



Funktionsweise iptables vs. nftables

- ```
1 payload load 4 offset network header + 16 => reg 1
2 compare reg 1 192.168.0.1
```

- ```
1 payload load 4 offset network header + 16 => reg 1
2 set lookup reg 1 load result in verdict register
3   { "192.168.0.1" : jump chain1,
4     "192.168.0.2" : drop,
5     "192.168.0.3" : jump chain2 }
```

iptables:

- iptables -A INPUT -p tcp -dport 22 -j LOG
- iptables -A INPUT -p tcp -dport 22 -j DROP

nftables: nft

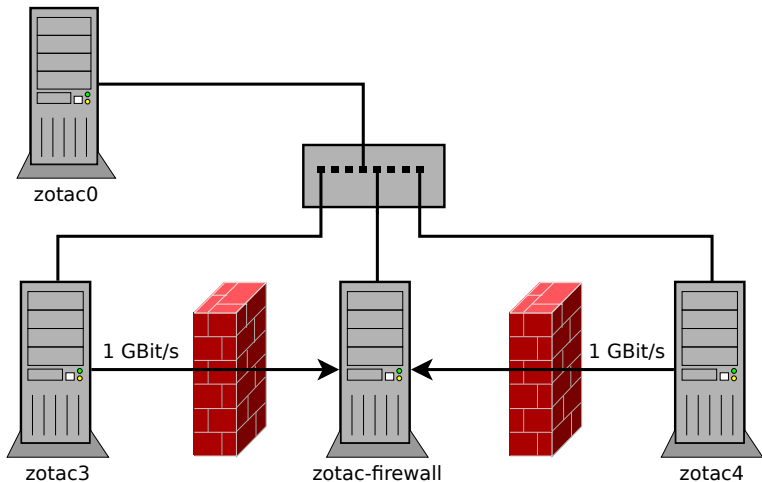
- nft add table filter
- nft add chain filter input "{ type filter hook input priority 0; }"
- nft add rule filter input tcp dport 22 log drop
- als Script:

```
1 #!/usr/bin/nft -f
2 table filter {
3   chain input {
4     type filter hook input priority 0;
5     ip protocol tcp dport 22 drop log
6   }
```

- nft bisher noch kaum in Linux-Distributionen verfügbar
- in Arch Linux bisher nur im AUR verfügbar

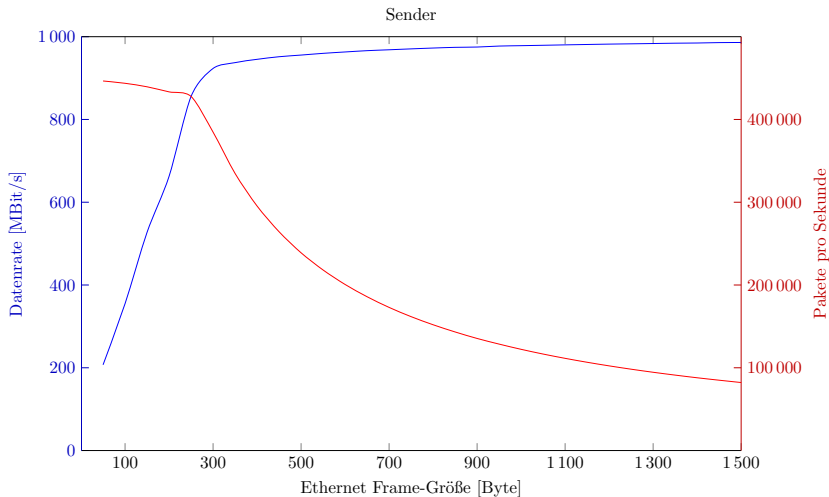
Performance-Vergleich

Testaufbau:

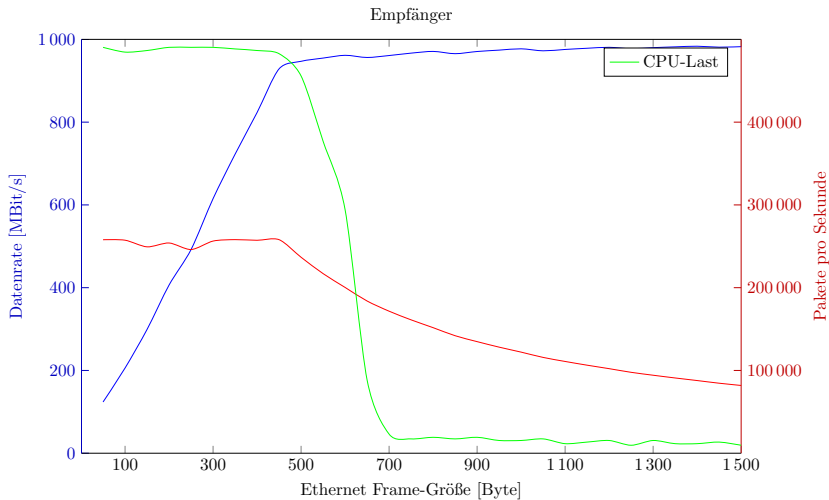


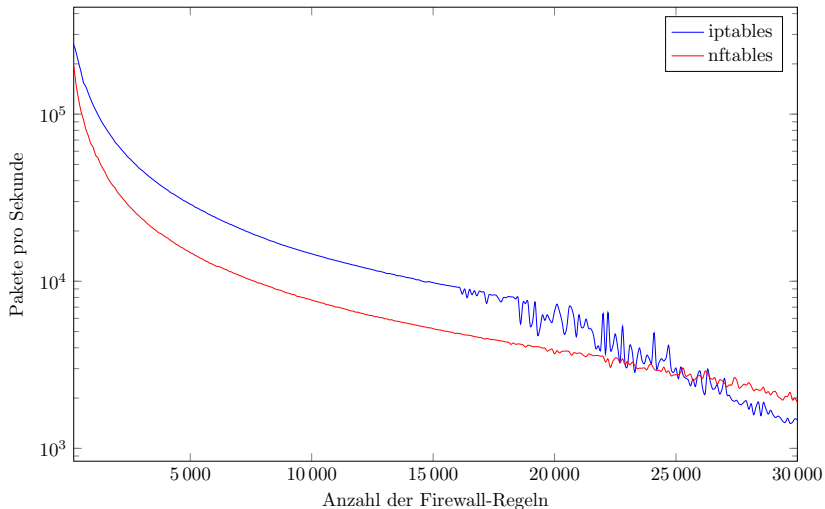
- Hardware:
 - Sender / Empfänger:
 - Intel Atom 330 (1,6 GHz)
 - NVIDIA MCP79 Ethernet Controller
 - 2 GB RAM
 - Firewall:
 - Intel Core 2 Duo E6750 (2,6 GHz)
 - Intel 82572EI und 82566DM-2 Ethernet Controller
 - 2 GB RAM
- Software:
 - Paket-Generator: pktgen
 - Netzwerk-Monitor: ifpps (aus netsniff-ng)
- Testablauf:
 - zotac3 sendet Pakete über zotac-firewall an zotac4
 - Firewall hat entsprechend viele Regeln
 - Empfänger verwirft Pakete noch im iptables-Stack

Ermittlung der optimalen Paketgröße

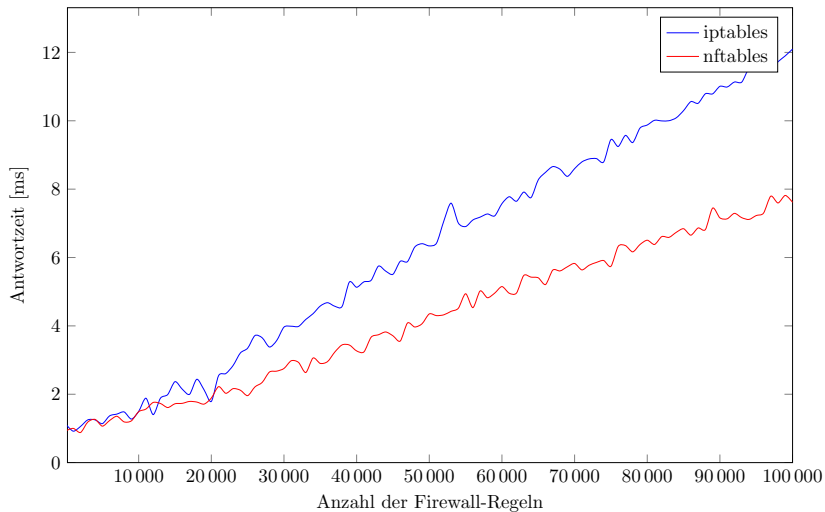


Ermittlung der optimalen Paketgröße





Verarbeitungsgeschwindigkeit / Antwortzeit



- iptables bezüglich Durchsatz noch überlegen, bei sehr vielen Regeln etwa gleich bzw. etwas schlechter als nftables
- nftables skaliert bezüglich Antwortzeit besser
- Vorteile nftables:
 - Konzept mit virtueller Maschine mächtig
 - leichte Erweiterbarkeit
- aber:
 - momentan bei mittlerer Regelanzahl noch wesentlich schlechter performant als iptables
 - bisher so gut wie keine Dokumentation verfügbar
- Fazit: (noch) nicht für den Produktiveinsatz bereit

- Projekt-Website von nftables:
<http://netfilter.org/projects/nftables/>
- Tutorial von Eric Leblond:
<https://home.regit.org/netfilter-en/nftables-quick-howto/>
- nftables Wiki:
<http://wiki.nftables.org/wiki-nftables/>
- Wikipedia-Artikel:
<http://en.wikipedia.org/wiki/Nftables>