Multi-Context-Firewalling mit Linux und nft

Inhalt

- Was war: iptables
- Was wird: nft
- Anwendungsbeispiel: Multi-Context-Firewall
- Demo

Was war?

- iptables (IPv4-Filterregeln)
- ip6tables (IPv6-Filterregeln)
- ebtables (Bridge-Filterregeln)
- arptables (Arp-Filterregeln)

=> Viele Tools für ähnliche Aufgaben

{ip,ip6,eb,arp}tables - Vorteile

- Bewährt u. zuverlässig
- Sehr umfangreich Dokumentation
- Erheblicher Funktionsumfang

iptables / ip6tables

- Administration der IPv4/IPv6 Filterregeln
- Wichtigstes Firewall-Tools
- Kann einzelne Filterregeln
  - Einfügen / anhängen
  - Ändern
  - Löschen

iptables - Konfiguration

```
#!/bin/bash
iptables -A FORWARD -p tcp --dport 80
  -d 139.11.6.13
  -j ACCEPT;
iptables -A FORWARD -p tcp --dport 443
  -d 139.11.6.13
  -j ACCEPT;
iptables -A FORWARD -p tcp --dport 22
  -d 139.11.6.13
  -s 10.1.30.0/24
  -j ACCEPT;
iptables -I FORWARD -s 114.3.0.0/16 -j DROP;
```

{ip,ip6,eb,arp}tables - Nachteile

- Wenig einsteigerfreundlich:
  - Mehrere Tools für ähnliche Aufgaben
  - Konfiguration kryptisch
  - Oft sehr (sehr!) viele Regeln notwendig
- Keine atomaren Updates

iptables - Probleme

- Sehr viele Regeln
- Schwer zu lesen
- Reihenfolge Config != Kern
- Modifikation:
  - Direkt durch iptables
    (Änderung der Config nicht vergessen!)
  - Änderung der Config
    (Aktivierung nicht atomar!)

Was wird: nft!

- Unterstützt seit Kernel 3.13
- Ersetzt {ip,ip6,eb,arp}tables
- Noch nicht in Distributionen ;-(

nft - Vorteile

- Viel einfachere Syntax
- Atomares Laden des Regelwerks
- Performanter

nft - Aufruf

- Kommandozeile (nft [command])
  - z.B.: nft list ruleset
- Interaktiv (nft -i)
- Aus Datei (nft -f)

nft - Regelwerk

```
#!/sbin/nft -f

table ip filter {
  chain dmz {
    # Regeln
  }
  chain lan {
    # Regeln
  }
  chain forward {
    # Regeln
  }
}
```

nft - Regel 1

```
define admin-net = 10.1.30.0/24

...

ip protocol tcp
  tcp dport ssh
  ip saddr $admin-net
  accept;
```

nft - Regel 2

```
define intranet-server = 10.17.0.3

...

ip protocol tcp
  tcp dport {http, https}
  ip daddr {
      www.example.com,
      $intranet-server,
      139.11.6.13
  }
  accept;
```

nft - Chain 1

```
chain dmz {
  tcp dport ssh
    ip saddr $admin-net
    accept;

  ip protocol tcp
    tcp dport {http, https}
    ip daddr {
      www.example.com,
      $intranet-server,
      139.11.6.13
    }
    accept;
}
```

nft - Chain 2

```
chain forward {
  type filter hook forward priority 0;

  ct state invalid drop;
  ct state established,related accept;

  ip protocol icmp
    icmp type echo-request
    iif {eth0, eth1}
    accept;

  jump dmz;
  jump lan;

  reject;
}
```

Multi-Context-Firewall

- Klassisch: Eine zentrale Firewall
- Problem: Sehr bürokratisch
- Lösung: Mandantenfähige Firewall
  - Eine physische Firewall
  - Mehrere logische Firewalls

Mandantenfähigkeit

- Mandant:
  - Kunde
  - Abteilung
  - Tochterunternehmen
  - Angeschlossene Behörde
  - ...
- Jeder Mandant hat eigenen Firewall-Context

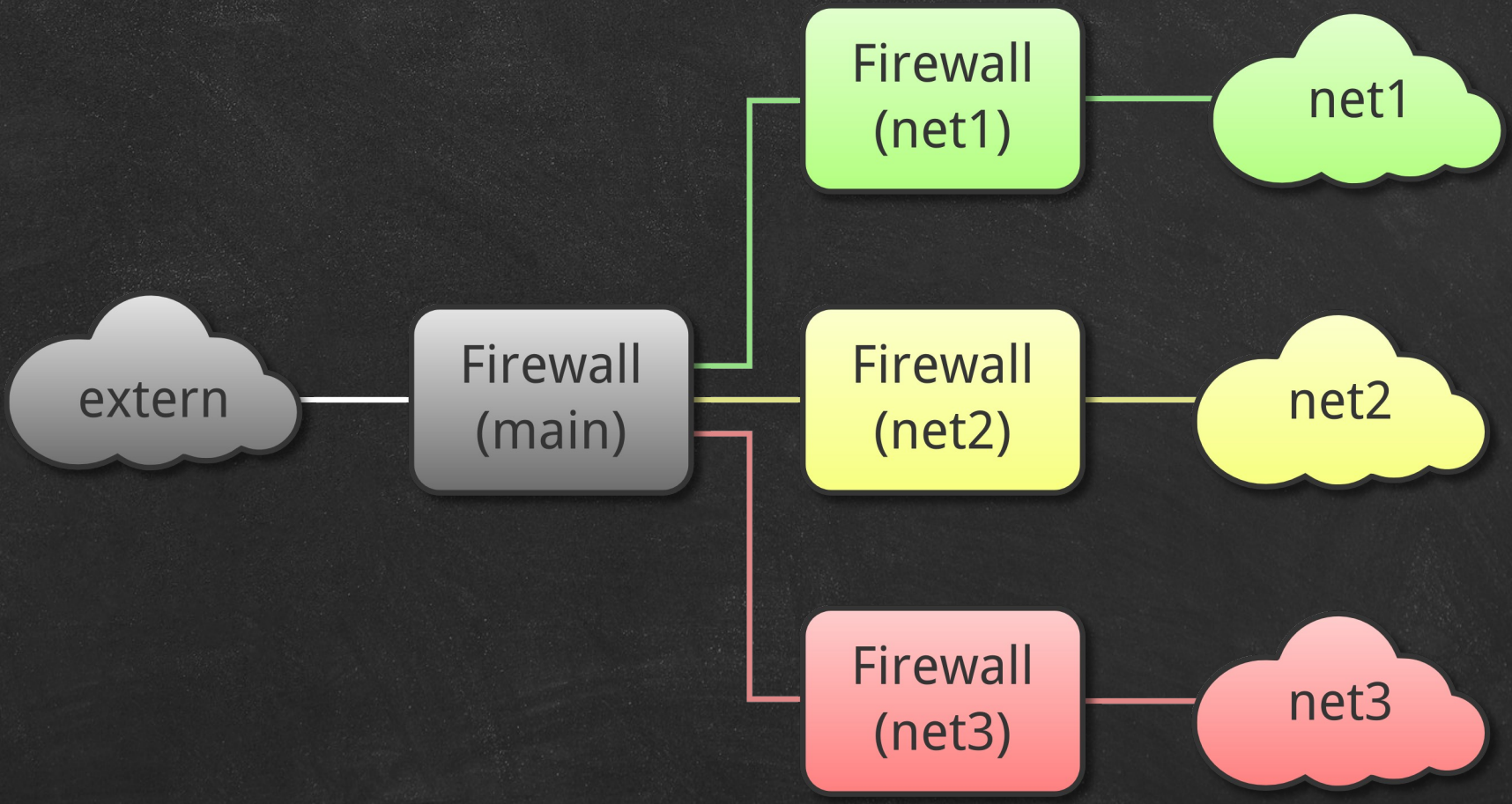
Technische Realisierung

- Netzwerk-Stack wird mehrfach instanziiert
- Jeder Stack umfasst:
  - Netzwerkkarten
  - Routingtabellen
  - Filtertabellen
  - ...
- Linux: Networknamespaces

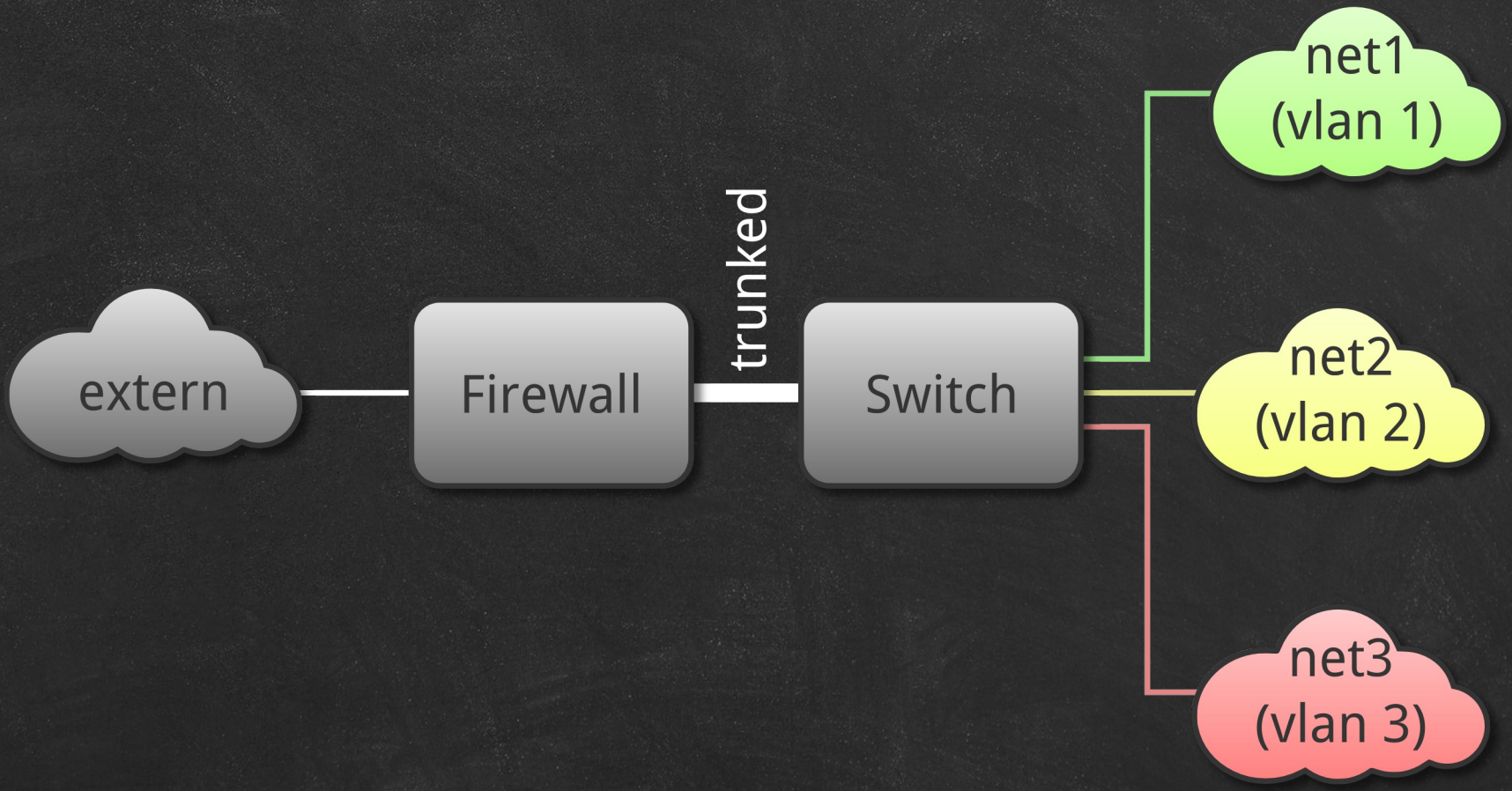
Networknamespaces

- Auflisten: ip netns list
- Erzeugen: ip netns add <NAME>
- Prozess starten:
  ip netns exec <NAME> <PROGRAM>
- Löschen: ip netns delete <NAME>
- Netzwerkkarten verschieben:
  ip link set dev <DEV> netns <NAME>

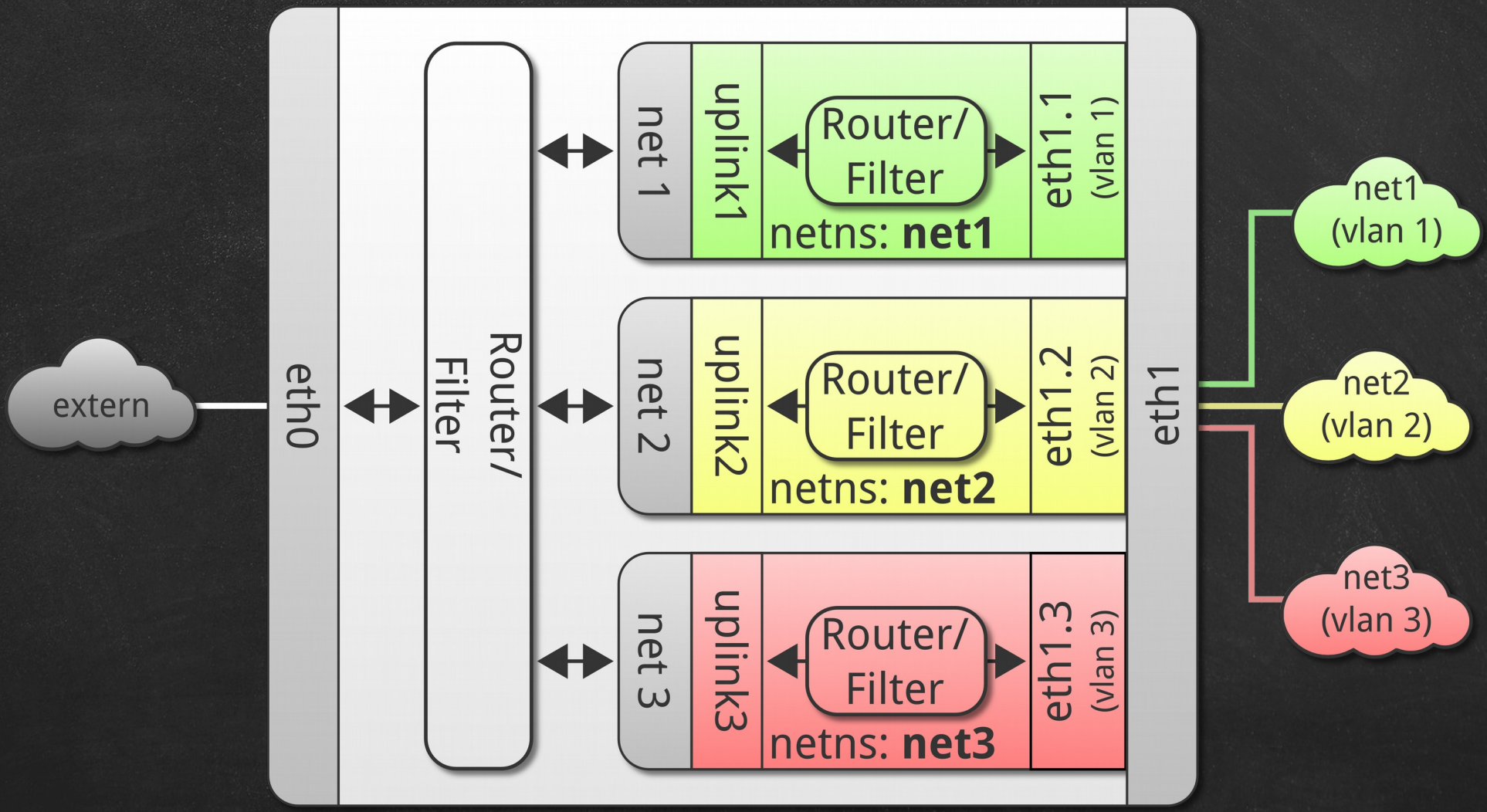
Logische Infrastruktur



Physische Infrastruktur



Firewall-Konfiguration



Demo!

Fragen?

Open Source Security Ralf Spenneberg
www.os-s.net
info@os-s.net