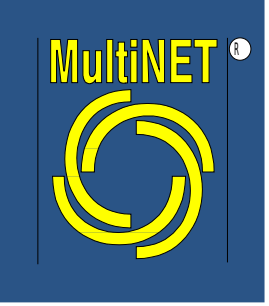


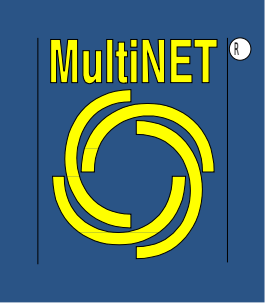
Eine hochverfügbare Firewall mit Linux-HA, `iptables` und `fwbuilder`

FROSCON, 23.8.2009

Dr. Michael Schwartzkopff

Eine einfache Firewall

- Eine einfache Firewall mit Linux ist schnell eingerichtet
  - 3 Schnittstellen (extern, intern, DMZ)
  - Prozessorleistung, Speicher und Festplatten sind heute fast ohne Bedeutung
- Probleme:
  - Hochverfügbarkeit: aktiv / passiv Cluster mit `pacemaker` *und* Synchronisieren des State Tables
    → kein Abreißen der Verbindungen im Fehlerfall
  - einfache Administration als Mittelweg zwischen Webinterface und eigenen Scripten



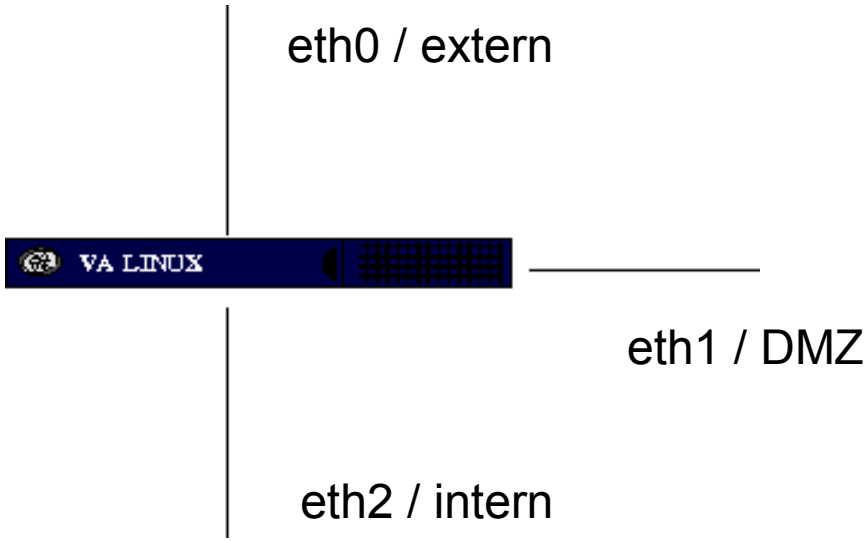
Administration mit `fwbuilder`

- Erzeugung eines Regelsatzes über ein Webinterface oder distributionseigene Werkzeuge zu unflexibel.
- Die Scripte selbst zu schreiben ist auf die Dauer zu umständlich. Besonders bei vielen Systemen.
- Lösung: `fwbuilder`
  - flexibel (!)
  - skaliert angemessen
  - GUI bietet Überblick

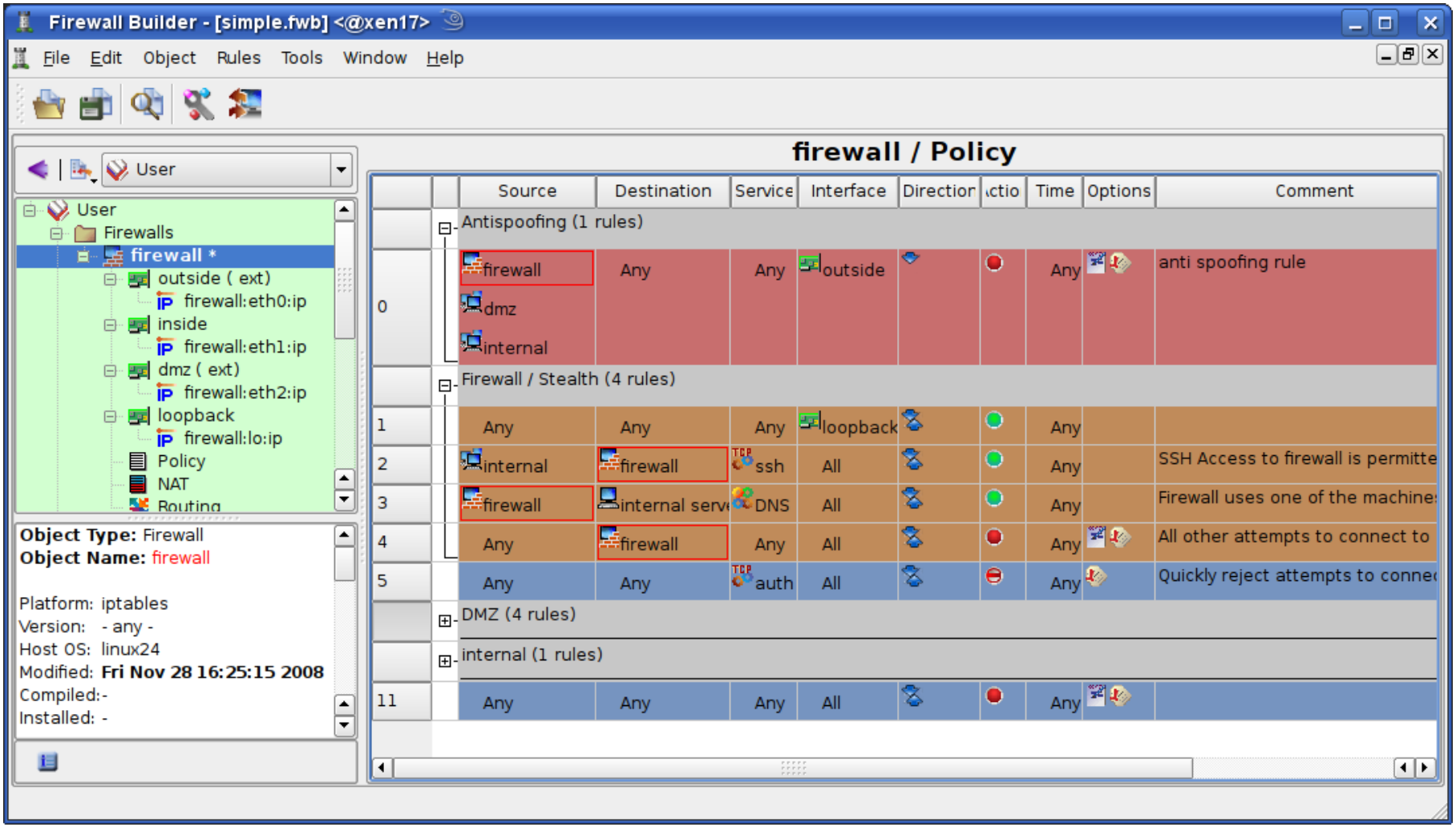
`fwbuilder`: Fähigkeiten

- Gut nutzbare GUI, die hinter kommerziellen Lösungen nicht zurückbleibt.
- Skaliert durch 2-Tier Architektur, d.h. Erzeugung des Regelsatzes und Durchsetzung sind getrennt.
- Die grafische Darstellung des Regelsatzes bietet einen bessern Überblick als Skripte.
- Automatische Entdeckung von Regelkonflikten.
- viel mehr, besonders in Version 3.

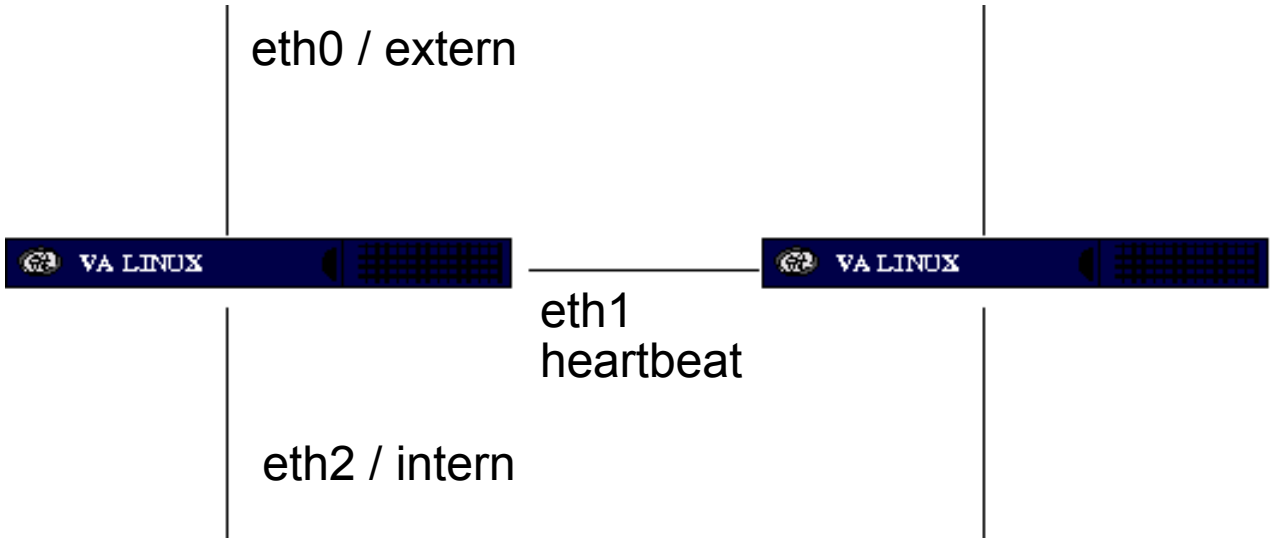
Einfache Firewall: Schema



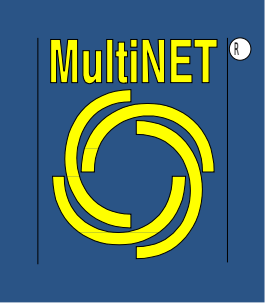
Regelsatz in fwbuilder



Bessere Firewall: Hochverfügbarkeit



DMZ optional



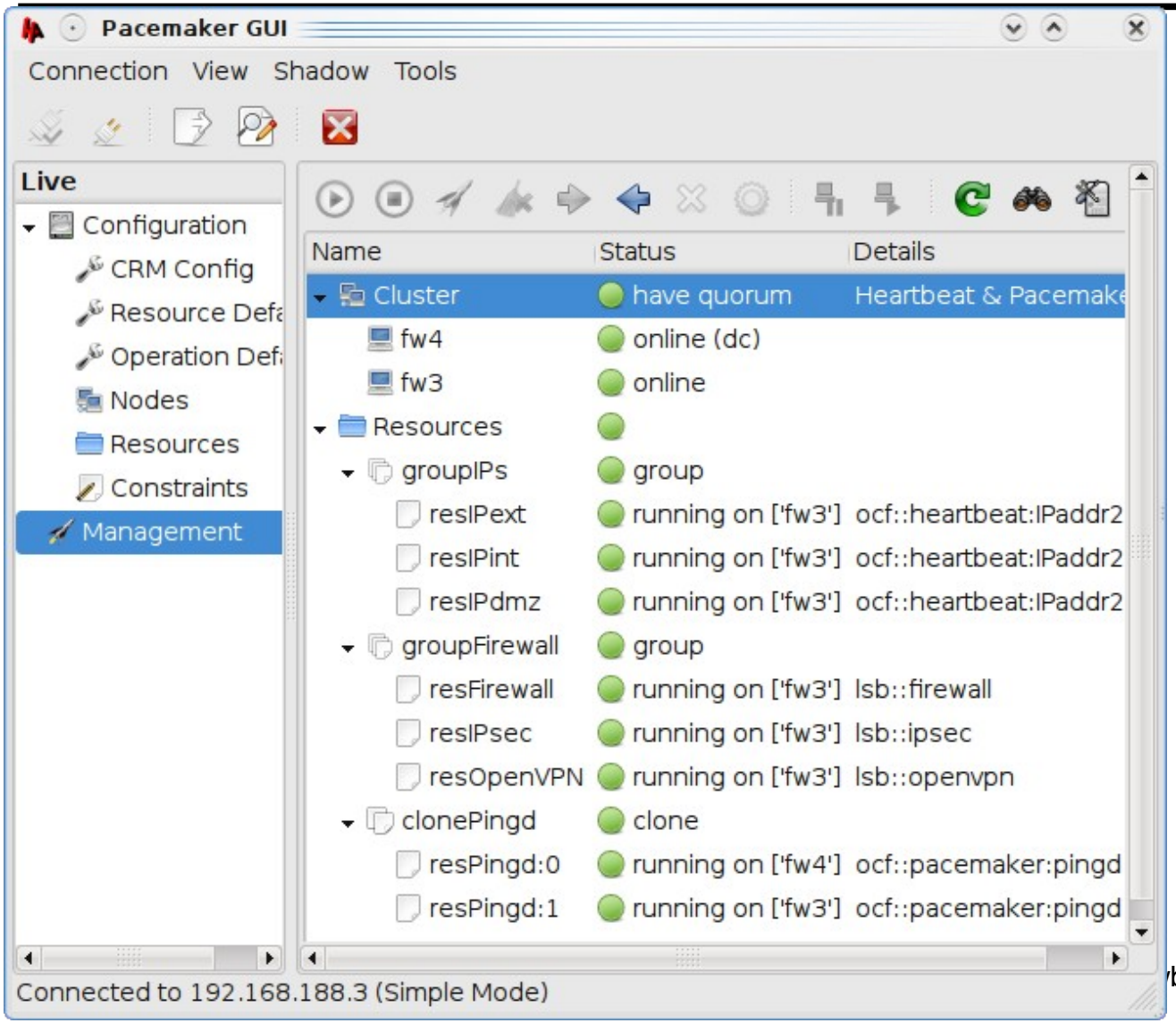
Aufbau einer HA-Firewall

- Bei Ausfall der aktiven Firewall übernimmt das passive System.
- Die Firewall als Gesamtsystem ist sofort wieder einsatzbereit.
- `pacemaker` *verwaltet virtuelle Cluster-Adressen*. Über diese läuft das Routing.
- `pacemaker` *kann* auch `ip_forward` verwalten. Muss aber nicht, weil keine Pakete an die passive Firewall weitergeleitet werden.

Ressourcen in `pacemaker`

- IP Adressen als `pacemaker`-eigene (OCF) Ressourcen.
  - Gute Integration in `pacemaker`
  - Ressource: `IPaddr2`
- Firewall als init-Script:
  - verwaltet `ip_forward`
  - ruft beim Start das Firewallscript auf
  - `status` überprüft den Zustand von `ip_forward`.
- Gruppierung der Ressourcen:
  - Co-Lokation auf einem Knoten und evtl. Anordnung

HA-Firewall in der GUI von `pacemaker`

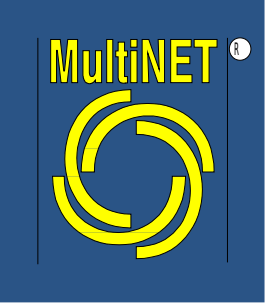


OpenAIS Kommunikation zulassen!

```
iptables -I INPUT -i eth1 -p udp -d 226.94.1.1
  --dport 5405 -j ACCEPT

iptables -I OUTPUT -o eth1 -p udp -d 226.94.1.1
  --dport 5405 -j ACCEPT
```

- Merken und später in den Regelsatz einbauen!



Synchronisation des Status beim Failover

- Beim Failover reißen alle Verbindungen ab, da der Connection Table zwischen den Knoten des Clusters nicht abgeglichen wird.
- Frühere Lösung `ct_sync` suboptimal.
- Jetzt: `conntrackd`
- Ist z.B. in Debian/testing (aka Lenny) enthalten.

```
# apt-get install conntrackd
```

- Achtung: Bei Lenny ist ein neuer Kernel notwendig.

conntrackd

- conntrackd verwaltet zwei Zwischenspeicher (einfacher Modus):
  - Intern: gibt den connection table des Kernel wieder. Dieser wird per Multicast veröffentlicht.
  - Extern: Meldungen von anderen Knoten werden hier zwischengespeichert, aber noch nicht an den Kernel weitergegeben
- Beim Umschalten wird der der Inhalt des externen Caches in die Tabelle des Kernels synchronisiert.
- Es gibt auch einen Modus, der direkt synchronisiert. Damit sind asymmetrische Systeme möglich.

/etc/conntrackd.conf (I)

```
Sync {
        Mode FTFW { ... }
        Multicast {
                IPv4_address 225.0.0.50
                IPv4_interface 192.168.1.2
                Interface eth1
                Group 3780
        }
}
(...)
General {
  Filter From Userspace {
    Protocol Accept {
      TCP
    }
    Address Ignore {
      IPv4_address 127.0.0.1
      IPv4_address (...) # cluster and dedicated IP addresses
    }
  }
}
```

Kommunikation für `conntrackd`

```
iptables -I INPUT -i eth1 -p udp -d 225.0.0.50
  --dport 3781 -j ACCEPT

iptables -I OUTPUT -o eth1 -p udp -d 225.0.0.50
  --dport 3781 -j ACCEPT
```

- Merken und später in den Regelsatz einbauen!

`/etc/init.d/firewall` anpassen

- Bisher:
  - *`start`* / *`stop`* lädt bzw. löscht Regelsatz.
  - `ip_forward` wird mit *`start`* / *`stop`* gesetzt bzw. gelöscht.
  - *`status`* fragt `ip_forward` ab.

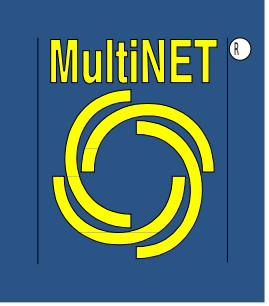
- Zusätzlich:
  - `conntrackd` Befehle (`-c`, `-f`, `-R`, `-B`) bei *`start`* zur Übernahme der externen Tabelle in den Kernel und Neuaufbau
  - `conntrackd` Befehl `-t` und `-n` bei *`stop`* zum Neuaufbau des externen Caches.

System fertig!

- Alle Voraussetzungen für eine HA Firewall fertig!
- Im Betrieb wird der *connection table* der aktiven Firewall immer in den Cache des passiven Systems übertragen.
- Download geht weiter, auch wenn die aktuell aktive Firewall abstürzt:
  - `pacemaker` erkennt den Fehler.
  - Die virtuellen Adressen werden auf den bisher passiven Knoten verschoben.
  - Dort wird auch der Regelsatz aktiviert und der *connection table* übernommen.

Integration in `fwbuilder`

- `fwbuilder` ist aktuell (Version 3.0) nicht für Cluster ausgelegt.
  - Anleitung unter *http://www.multinet.de/HAFirewall*
- Neue Fähigkeiten in Version 3.1:
  - Cluster State Sync: `conntrackd`
  - Cluster Schnittstellen: VRRP, OpenAIS
  - Automatische bond und VLAN Schnittstellen
- Version 3.1 zur Zeit noch im pre-beta Stadium.



Demo: `fwbuilder`-3.1

- Live Demo von `fwbuilder`-3.1
- Regelsatz: froscon.fwb

Tipps und Tricks

- Auf NAT achten, besonders bei der Multicast Adresse 225.0.0.50.
- Testen, testen, testen.
- Failover geht nicht von alleine. `pacemaker` muss schon entsprechen konfiguriert werden!
  - Pingknoten mit Bedingungen für einen Ausfall
  - Überwachung von Systemressourcen mit Bedingungen.
  - Überwachung von Ressourcen und Reaktion auf Fehler.
- Überwachung und Berichte einrichten. Nur ein überwachter Cluster wird funktionieren!

Ein detaillierteres, ein bisschen veraltetes HOWTO für `fwbuilder` in Version 3.0 liegt hier vorne aus oder findet sich auf

http://www.multinet.de/HAFirewall

Danke für die Aufmerksamkeit!

Fragen?