Transparentes Load-Balancing für Network Intrusion Detection Systeme

Matthias Vallentin & Robin Sommer

Motivation

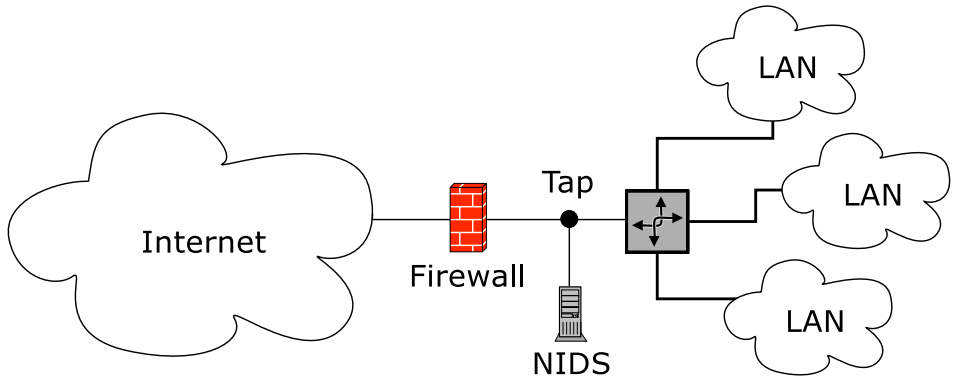
- High-Speed Umgebungen stellen neue Herausforderungen dar
  - Backbones, Universitäten
- Network Intrusion Detection Systeme (NIDS) stoßen an ihre Grenzen
  - CPU, Packet Capturing
- bisherige Ansätze:
  - proprietäres NIDS auf teurem Hardware-Cluster
  - Load-Balancing mit gering kooperierenden NIDS auf PC-Hardware
- Kombination der Vorteile: Cluster aus PC-Hardware.

Outline

- Network Intrusion Detection
- Transparentes Load-Balancing
- Bro - Open-Source NIDS
- Bro Cluster
  - Hardware
  - Konvertierung der Bro Policy-Skripte
  - Evaluation
  - Einsatz in Produktionsumgebungen

Network Intrusion Detection

- ein Network Intrusion Detection System (NIDS) versucht Verletzungen der Netzwerk-Sicherheitsrichtlinien zu erkennen
- passiv vs. aktiv
- Erkennungsmethoden
  - misuse detection
  - anomaly detection
  - specification-based detection
- State
  - Abbild des Netzwerkzustandes
  - Bsp: aktive Verbindungen, Scan Detector



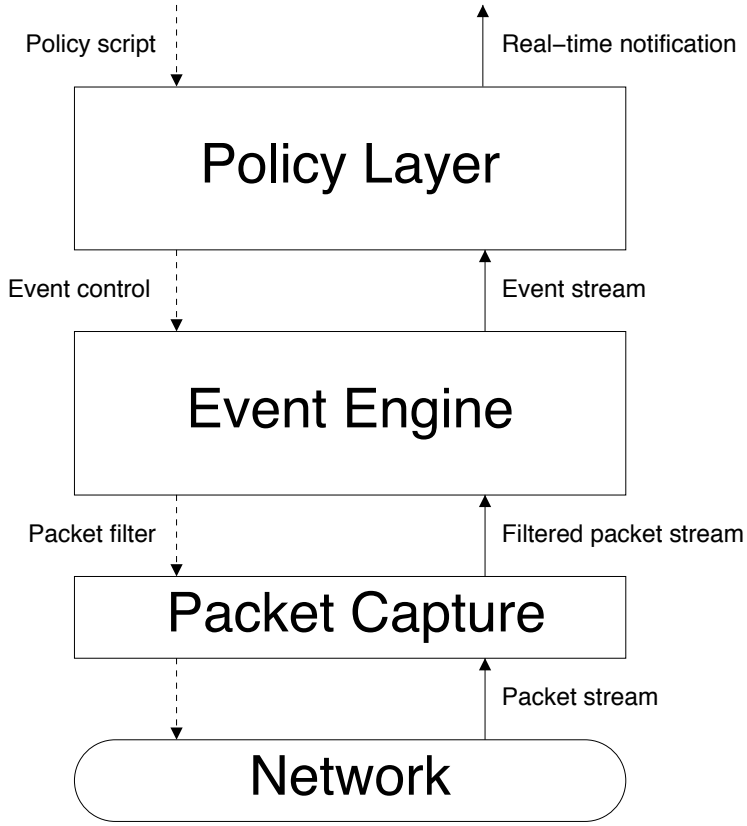
Transparentes Load-Balancing

- akutes Problem am Lawrence Berkeley National Laboratory (LBNL):
  - 15 seperat administrierte NIDS (Bro)
  - aufwendige manuelle Log-Auswertung
  - kein kohärenter, transparenter Gesamtüberblick
- operativer Wunsch
  - ein transparentes NIDS, hinter dem sich beliebig viele *Nodes* verbergen
- Lösungsansatz
  - NIDS Cluster: Verbreitung von State unter den Nodes
  - zentrale Log-Aggregation

Bro - Open-Source NIDS

- High-speed, large volume monitoring
- komplex, aber sehr flexibel

- Packet Capturing: *libpcap*
  - plattformunabhängig
- Event Engine (core)
  - policy-neutral
- Policy Layer
  - Bro Script Sprache



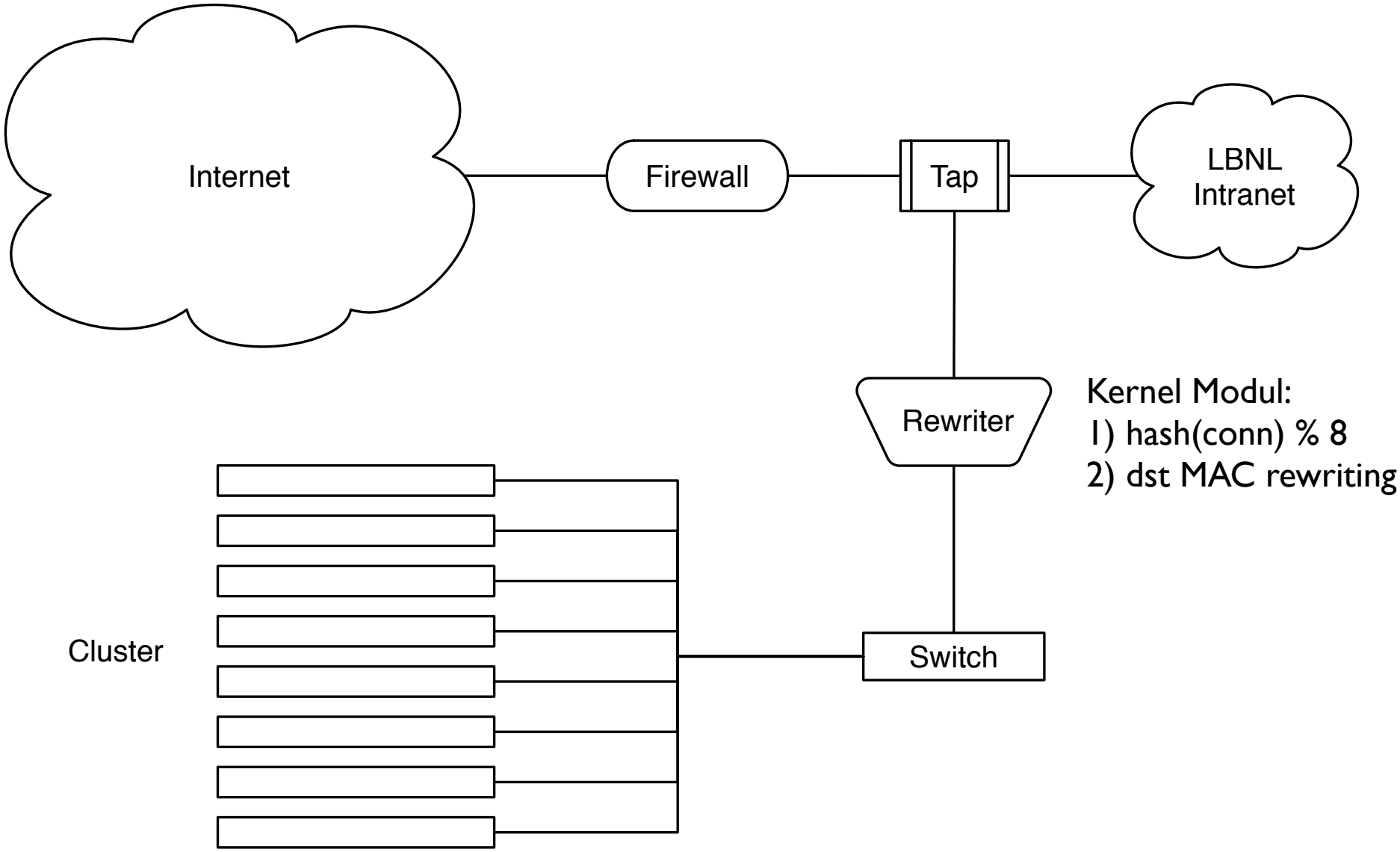
State Synchronisation

- Korrelation/Aggregation von Logs meist trivial, aber unzureichend
  - `"Notice: Host A scanned host X"`
  - `"Notice: Host A scanned host Y"`
- Stattdessen Analyse selbst, nicht die Ergebnisse verteilen
- Synchronisierung von policy-neutrale State über mehrere Nodes ist nicht einfach
  - Bsp: Bro Scan Detector: user policy script (¬ core)
    - verwendet Tables, die über Verbindungsgrenzen hinweg State halten, z.b. `distinct_ports[addr][addr]`
    - Veränderungen an solchen Tables werden propagiert
  - *Independent State* [Sommer05]

Bro Cluster

1. Hardware: Infrastruktur des Clusters
2. Konvertierung der Bro Policy-Skripte
3. Evaluation
   - 3.1. Korrektheit
   - 3.2. Performance
4. Einsatz in Produktionsumgebungen

Hardware: LBNL



Konvertierung der Policy-Skripte

- Propagieren von synchronen Variablen problematisch
  - `if ( ++foo_cnt == treshhold ) {...}`
- Sicherstellung der Integrität lokaler Mengen
  - `+=` Operator für Mengen: Attribut `&mergeable`
  - `local foo: set[addr] &mergeable`

Evaluation

- Korrektheit
  - Erkennungsrate: einzelnes Bro ↔ Cluster
- Performance
  - Skalierbarkeit: linear?
    - Erweiterbarkeit
    - Transparenz
    - größerer Cluster? ~100 nodes? NERSC Center!
- Methode: *Pseudo-Realtime* [Sommer05]
  - Arbeit mit Traces: Reproduzierbarkeit
  - simuliert Echtzeit-Traffic

Einsatz in Produktionsumgebungen

- LBNL
  - primäre Entwicklungsumgebung
- nächster Cluster im Live-Betrieb:
  - UC Berkeley Backbone
  - Münchner Wissenschafts Netz (MWN)

FIN

[Sommer05] Robin Sommer, Vern Paxson. Exploiting Independent State For Network Intrusion Detection. In *Proceedings of the Annual Computer Security Applications Conference.* 2005.