Network-Wide Policy Enforcement

Michael K. Reiter
(joint work with V. Sekar, R. Krishnaswamy, A. Gupta)
Enforcing Policy in Future Networks

• MF vision includes enforcement of rich policies “in the network”
• E.g., intentional data receipt
  – Only if (or if not) from these addresses
  – Only if a “well-formed” request
  – Only if it’s not malicious
  – …
• How can we scale rich policy enforcement?
  – Use modern intrusion detection/prevention policies as examples to drive the research
Scaling intrusion detection & prevention

Traffic increase → More packets to process
User applications changing → More functions/modules
Attacks evolving → Larger set of “rules” to apply
Traditional approaches for scaling NIDS/NIPS

1. Build clusters
2. Better algorithms
....

Single-vantage-point solutions designed for *perimeter* deployments (gateway between internal/external network)
Single-vantage view is restrictive

Capacity of each installation = 50 units

Overloaded!

Offload to N3?

Many NIDS/NIPS infrastructures are multi-node deployments e.g., ISPs, large enterprises, datacenters
Alternative: Network-wide scaling

Treat NIDS/NIPS infrastructure as one system
→ Leverage on-path opportunities for distributing work
→ Complements single-vantage scaling
Key requirements

1. Resource Constraints
2. Placement Affinity
3. Network-wide objectives

Systematic designs for network-wide NIDS/NIPS deployment
Outline

• Motivation for network-wide approach
• NIDS deployment
• NIPS deployment
• Discussion and summary
System model

Provide functionality equivalent to a single NIDS on all traffic
Minimize the maximum load (CPU/Memory) across the network
Identifying candidate nodes for offloading

Partition traffic spatially; Identify a set of nodes for each partition
Any node in that set can provide the NIDS function \( \rightarrow \) Coordination Unit

Signature
Partition1: \{N1,N5,N3\}
Signature
Partition2: \{N2,N5,N4\}
Identifying candidate nodes for offloading

Partition traffic spatially; Identify a set of nodes for each partition
Any node in that set can provide the NIDS function → Coordination Unit

Scan Partition 1: \{N1\}
Scan Partition 2: \{N2\}
Scan Partition 3: \{N3\}
Scan Partition 4: \{N4\}
System model

Provide functionality equivalent to a single NIDS on all traffic
Minimize the maximum load (CPU/Memory) across the network

Control fraction of traffic each node processes per module/coordination unit
NIDS optimization framework

**Inputs:**

- **Network information**
  1. Coordination units
  2. Traffic specification
  3. Routing information

- **Resource footprints**
  Mem/CPU profile for each NIDS module

- **Node capacities**
  Memory/CPU

**Objective:**

- Provide equivalent coverage
- Minimize maximum load

**Network-wide optimization using linear programming**

- Fraction of traffic each node processes per NIDS module
- Non-overlapping hash ranges per coordination unit
Prototype implementation in Bro

**Hash check:**
Compute $\text{hash}(\text{header})$
If $\text{hash}$ in range assigned for module
  Process the packet
Else
  Ignore packet

Raw packet stream → Traffic

Filtered event stream

Early drop; More efficient!

Easy to implement; Some modules still need checks here
Evaluation: CPU scaling

Emulated Abilene topology: 100,000 sessions; gravity-model matrix
Result shows maximum processing footprint across 11 network nodes

Reduces maximum processing footprint by 50%!
Outline

• Motivation for network-wide approach

• NIDS deployment

• NIPS deployment

• Discussion and summary
System model

Payload signature, ACL, Firewall

Set of *rules*
Drop traffic that matches the *rules*
Any node on path can apply the rule

Rule capacity constraints (in addition to CPU, memory)
NIPS optimization framework

Inputs:
- Network information
  - Traffic, Routing
- NIPS rules
  - Rule match rates
- Resource footprints
  - Mem/CPU profile
  - TCAM per rule
- Node capacities
  - Mem/Processing capacity
  - TCAM capacity

Objective:
Reduce footprint of unwanted traffic = (match * volume)

What rules to enable on each node (TCAM constraint)

Fraction of traffic to apply enabled rules on (Mem/CPU constraint)

NP-hard! “Enable” introduces binary variables
Becomes an Integer linear Program

“Enable” introduces binary variables
Becomes an Integer linear Program
Performance of our approx. algorithm

Different PoP-level topologies from Rocketfuel

Achieves more than 92% of LP-upperbound (and hence OPT)
Other issues

• Providing redundant coverage for NIDS
• Making NIPS robust to adversarial evasion

• Network dynamics
  – Time to recompute optimal solutions
  – Conservative traffic inputs
  – Correctness under routing changes

• Provisioning/upgrades
Conclusions

• Exploit spatial opportunities for load balancing
  – Complements ongoing work in better single-vantage-point solutions
• Key issues:
  Resource constraints, placement affinity, network-wide goals

• Best addressed using network-wide coordinated approach

• NIDS deployment
  – Formulation as a linear program
  – Network-wide NIDS prototype in Bro
• NIPS deployment
  – Formulation as a Integer Linear program
  – Rounding based approximation algorithm