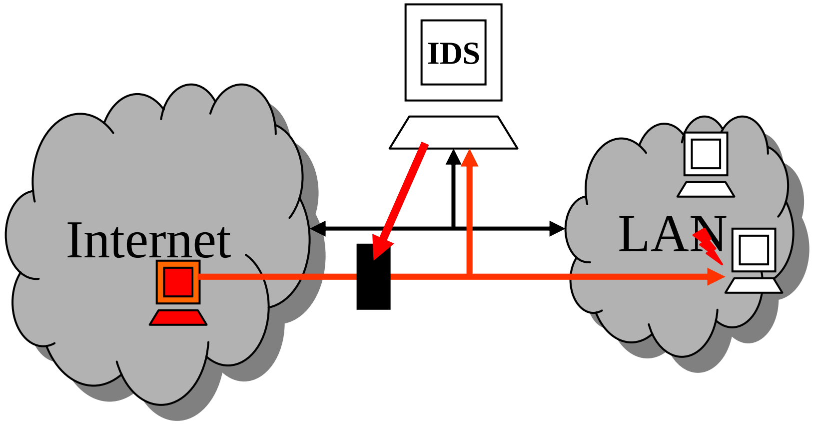
Shunting: Detecting and Blocking Network Attacks at Ultra-High Speeds

Vern Paxson
International Computer Science Institute
&
Lawrence Berkeley National Laboratory

Nicholas Weaver
International Computer Science Institute

José María González
UC Berkeley

The Problem:
Intrusion Detection and Response



- Network Intrusion Detection System
  - Monitors the traffic between the local network and the Internet
  - Attempts to detect and log attacks
- Network Intrusion Prevention System
  - IDS now tries to block attacks in progress
    - Before the victim can be compromised
  - Also block subsequent attacks from an offending system
- Terms are often used interchangeably
  - We will use the term IDS to describe both goals: logging of attacks and blocking attacks as they occur
- We want IDSs because they allow a "Default Allow" policy rather than a "Default Deny"
  - Default allow enables greater collaboration, which is vital for scientific research mission

We Need Some Enhancements

- Unfortunately, the current state-of-the-art is limited
- Need better scalability:
  - Techniques for 1 Gbps (prototyping and general deployment)
    - Ideally low cost deployment
  - Scalable to 40 Gbps
    - 40 Gbps networks are "special": Flows are dominated by traffic the IDS is not interested in, such as large experimental dataset transfers
    - 40 Gbps is way beyond the state of the art
- Need inline operation:
  - All traffic passes through the IDS
    - The IDS, not router ACLs, can block individual sources or terminate individual connections
      - Router-based blocking has scalability problems in the face of 1000s to 10,000s of hostile systems attack a network
      - Router-based blocking can’t effectively halt an attack-in-progress
      - Need to drop without burdening security administrators
  - Easy to do for 100 Mbps, but very challenging for 1 Gbps+
- Leverage the Bro Intrusion Detection Systems
  - Already deployed at multiple, very-high-performance open facilities
  - High level policy-based analysis engine

Key Observation:
For an IDS, most traffic is uninteresting

- All the traffic needs to flow ***through*** the IDS…
  But most of the traffic doesn't need to be examined ***by*** the IDS
  - Scientific research traffic volume is dominated by a few very-large bulk transfers
  - Encrypted connections (ssh, https/ssl, vpn connections)
    - At least after the connections are properly setup
- Yet the IDS must see all connections as they occur
  - Even for encrypted data, session initiation needs to be monitored
    - And when to stop monitoring may be very protocol specific
  - Thus we can't use a simple ACL in a router to redirect some traffic through the IDS
  - And the IDS must block known sources of bad traffic
- Idea: Allow the IDS to control what it sees
  - Using a programmable network filter element
    - Filter element can also handle blocks
  - Develop policies which can utilize this element

We Are Developing Both Mechanism and Policy

- Mechanism: The Shunt
  - A hardware device to allow the IDS to control what it sees
- Key Idea: Table-based decisions
  - Every packet header is looked up in a series of tables
    - Highest priority match is used
  - Packet is then directed either to the IDS, to the destination, or dropped
- Mechanism is "hardware friendly"
  - No per-byte work or pointer chasing
  - Fixed memory amounts and access
  - Limited queueing required
- Prototype designed for scalability
  - Targeting 1 Gbps but ensuring scalability
    - Limited memory access
    - Small & narrow FPGA datapaths
- Policy: Shunting
  - How to use a shunt to enhance the Bro IDS
    - Both with and without the presence of a hardware shunt
- Software for managing a shunt
  - Understands a Shunt's capacity limitations
  - Provides an "unlimited" abstraction to Bro
- Enhancing policy scripts with shunt control
- Evaluating what traffic must be seen by the IDS and what can be ignored
- Ensuring that the mechanism is useful
  - Policy must dictate mechanism's capabilities

The Mechanism:
The Shunt

- The ***Shunt***: A device which an IDS can use to control what it sees
  - For a given packet, either:
    - Forward the packet onward toward the destination
    - Drop the packet
    - Sample/mirror: Send the packet on and send a copy to the IDS
    - Divert: Forward the packet to the IDS
      - The IDS then decides whether to forward the packet or not
- The Shunt's mechanism: Fixed-address table lookup
  - Based on the packet headers, look in several tables to find the action
    - If no entries, use a default behavior: Divert the packet to the IDS
  - Easy to implement both in software and hardware
    - Fixed small memories: A few MB
    - Fixed memory access patterns: <5 memory accesses per packet

The Shunt's Mechanism

- For each packet, extract the IP/TCP headers
  - Look up the source and destination IPs in the IP_table
  - Look up the connection tuple in the Connection_table
  - Look up the IP/TCP flags in the Flags_table
  - The IDS can change the entries in all the tables
- Each entry (if valid) has an action and priority
  - Forward: Pass the packet onward without notifying the IDS
  - Drop: Block the packet without notification
  - Sample: With probability ***P***, send a copy to the IDS
  - Divert (Default): Pass the packet to the IDS for decision
    - The IDS then decides whether to forward the packet or drop it
- This allows the IDS to control what it sees
  - Block hosts which are known to be malicious
  - The IDS can examine and block all possibly dangerous traffic
  - While allows connections which the IDS has concluded are "safe" to not bother the IDS anymore
    - Add an entry into the connection table

Single-Sided Error in the Shunt

- The shunt's tables may be incomplete
  - In order to bound the memory access and total memory requirements, the tables will need to evict entries due to lack of space
    - Even for software, fixed access patterns offer performance advantages
- Default shunting behavior limits the damage
  - A falsely evicted entry returns to default-shunt
  - Evictions in the IP or connection table must also have a corresponding eviction in the connection table
    - Don't want to cause ordering problems by having conflicting evictions
- Thus the shunt doesn't require 100% accuracy
  - Rather, it is an approximate device which relies on the IDS system to maintain full state
  - Errors are "safe", any bad eviction decision by the Shunt gets reviewed by the IDS

Other Experiences
With Hardware Development

- On previous board, verified:
  - Gbps operation of key components
    - Multiple Gigabit Ethernets at line rate
    - Customized FIFOs
    - Header extraction
    - Address-encryption for table lookups
    - All components at 125 MHz/8b datapath -> GigE line rate
    - Easily fits within hardware budget
- Very low latency architecture:
  - For forwarded packets, latency measured in nanoseconds
    - Assume that a packet is "good"
    - Packets which must be shunted or dropped are halted by declaring an ***underrun*** to the Ethernet MAC:
      The packet is corrupted and ignored by the recipient
      - Don't need to receive entire packet before sending it on

The Shunt:
Status

- Software implementation running and stable, coupled with Bro
  - Been running for multiple weeks on real traffic in ICSI's LAN
    - So all packets for several systems are passing through the Shunt coupled to Bro
  - Running on multiple hour traces of LBNL traffic
- Hardware implementation delayed:
  - Switching to a new platform: ***NetFPGA2***
  - New board offers 4xGigE, 4 MB SRAM, PCI interface, coupled to an FPGA
  - Several advantages:
    - Better support
      - Including net drivers
    - Tighter coupling with host
      - Simplifies design
      - Able to prototype multi-board solution for scalability
      - 10-GigE boards have a similar interface
    - Much lower cost
      - Can deploy more instances

Shunting:
Policies using a Shunt

- Just having a mechanism is insufficient: need policies to ***use*** the shunt
- Bro has been modified to allow scripts to change the state of the shunt for a particular connection
- We are constructing analyzers to use this mechanism
  - E.G. the **ssh** analyzer observes connection setup, but then will cut through the remaining traffic once the connection is well established

How Effective Is Shunting?

- Some testing:
  - A ~2 hour, ~1.2M connections, ~127M packet, ~113 GB trace of LBNL running through Bro with a Shunt
    - Only some Bro policies take advantage of shunting
- Only 25% of the packets and 19% of the data needs to be examined by the IDS
  - For the traffic examined by Bro, 80% of the bytes were HTTP traffic
    - Large data transfers for images/files
  - 1/2 of the remaining traffic were three protocols (SMTP, HTTPS, IMAP/SSL) where the analyzers didn't include shunting capability
- Evaluating mechanisms to reduce this traffic further
  - A "length" field on the connection table: Forward this connection until ***L*** bytes have been transmitted
    - Would allow the IDS to bypass larger items within a connections

Summary

- The Shunt, a new mechanism for IDS:
  - A mechanism for the IDS to control what traffic it examines
    - And the ability to block any offending traffic
  - Software implementation: Tested and stable, running for months
  - Hardware implementation: 1Gbps prototype in development
    - Algorithms scale to 10-40 Gbps applications
- Shunting, policies using a Shunt:
  - Effective at greatly reducing the traffic through the IDS
    - Current policies reduce traffic load by >75%
      - Some improved policies should reduce the load by >90%
  - Allows Bro to act as an IPS
    - Detect and block attacks
  - Tested in the real world
    - Active deployment in ICSI's LAN
    - Off-line testing using traces of LBNL's access link
      - Test deployment at LBNL in ~2 months