

Worms vs Perimeters: The Case for Hard-LANs

**Nicholas
Weaver**
International
Computer Science
Institute

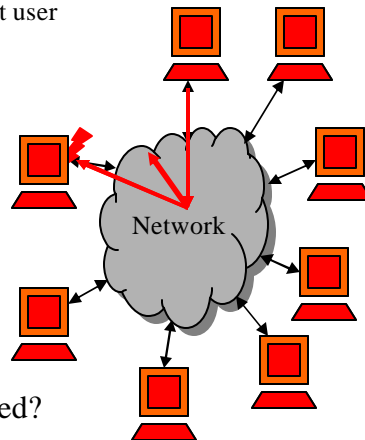
**Dan
Ellis**
The MITRE
Corporation

**Stuart
Staniford**
Nevis Networks

**Vern
Paxson**
International
Computer Science
Institute

What Are Worms?

- Worms are self propagating malicious programs
 - Spread from system to system without user interaction
 - Previous worms have infected >8 million systems in a single attack
 - Disrupted systems have included nuclear power plant monitoring, financial networks, 911 systems
- Reactive defenses *must* be automatic
 - "Internet Quarantine"
Moore et al
- But *where* should defenses be placed?

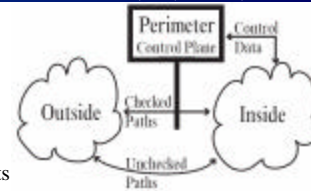


Defending the Enterprise

- We are interested in defending enterprise networks
 - Operationally critical systems for the US economy
 - Many techniques could then be adapted to retail ISPs
 - People will pay to defend themselves, not others
 - Internet suffers from the "least common denominator" problem
- Answering the question: where to place defenses?
 - How perimeters operate
 - How coarse-grained perimeters (firewalls) fails
 - How fine-grained perimeters (anti-virus) fails
- What we believe is necessary: Hard-LANs
 - Building defenses into the LAN fabric
 - Either in the switches, the NICs, or as separate devices
 - What do we believe needs to happen to cost/performance?
 - What do we believe needs to happen to false-positives?

What is a Perimeter?

- A perimeter is a control element in the network
 - It can monitor, modify and block traffic on the checked paths
 - It can't observe data on the unchecked paths
 - The control plane protects the logic of the perimeter
 - The perimeter can communicate with other components of the defense
- Some perimeter properties
 - A single penetration means that *this* perimeter can't offer any guarantees about inside state, but can still observe and block subsequent traffic
 - A penetration of the control plane and no more assertions can be made
 - As the coverage increases, the likelihood of a breach never decreases and often increases
 - Perimeters are composed to form an overall posture
 - Defense in depth attempts to compose perimeters to strengthen the overall posture
 - Containment attempts to limit damage from perimeter breaches
 - Accurate (low false positive) perimeters are generally more composable than sensitive (low false negative) perimeters



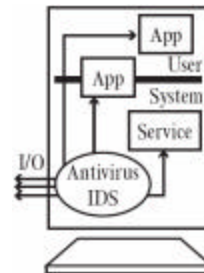
How Coarse-Grained Perimeters Fail

- Coarse-grained perimeters (Firewalls and NIDS) have been generally adequate against previous threats, but fail against worms
- Worms entering through the firewall
 - VPNs from affiliated institutions
 - Often undergo less scrutiny
 - Client-side (IE or email) attacks
 - Weaknesses in the static defense
- Worms avoiding the firewall
 - Can even cross "air-gapped" networks
 - Notebooks, mobile media
 - Wireless access points
 - Including rogue APs generated by "helpful" notebooks
- It only takes a single failure for the worm to enter the network
 - Unchecked paths continue to proliferate
 - Internal firewalls help but may not be enough
 - Such internal perimeters still tend to be very coarse-grained



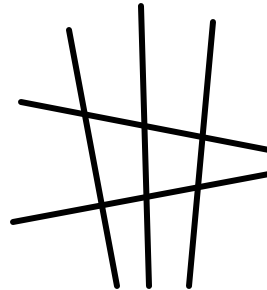
How Fine-Grained Perimeters Fail

- End host IDS/Firewall/AV can be effective at preventing infection
 - Has complete visibility over all traffic
- Often difficult to manage
 - Software on every end-host
 - Different types of end-hosts require different software
- Suffers from brittle failure
 - Once infected, all bets are off
 - Control plane is usually easy to penetrate when the system is compromised
 - Many proposed worm-defenses are cooperative
 - Use infected machines as a detection mechanism to warn other systems
 - Other defenses rely on containment
 - Limit the damage an already infected machine can do
 - Thus brittle failure limits the roll possible in worm defense:
 - Can only provide a useful defensive service before infection occurs



Our Proposal: Place the Defenses in the LAN

- Break the network into pieces
- Not on the end hosts
 - Avoid brittle failure
- But located close to the end systems
 - Limit the damage from breaches to a single area
 - Limit the number of unchecked paths to each area
 - Extensive visibility into the network traffic
- Thus placing defenses in the LAN
 - In the switches themselves as additional functionality
 - As separate devices in the LAN
 - NIDS on switch uplinks
 - On the NICs outside of the host's control (3-COM "Embedded Firewall" Ethernet cards)
 - If TCPA techniques ever work, on the end hosts
- This will require significant research to address the challenges involved, beyond just worm-detection
 - Cost/Performance needs to radically improve compared with conventional NIDS
 - False positive rate also needs a drastic reduction



Hard-LANs and Cost/Performance (1)

- We use firewall/NIDS as the base-case for evaluating cost/performance
 - It's a system we generally understand
 - Firewall/NIDS are network devices
 - Different design space when compared to end-host systems
 - They represent what companies will "reasonably spend" to protect themselves today
- Data rates in the LAN are much greater
 - NIDS: a T3 is ~45 Mbps, at ~\$2000/month
 - LAN: A 24 port, Gbps Ethernet switch is \$1500
 - Internal bisection bandwidth can exceed >20 Gbps
 - Even if underutilized, bursts may require supplying full-bandwidth
 - So the LAN can easily support 1 order of magnitude greater traffic volume

Hard-LANs and Cost/Performance (2)

- The number of devices also increases drastically
 - Internet connection: one or two devices (for redundancy)
 - Hard-LAN: one device per switch? In each switch? On each tap?
- If system cost is to remain comparable, cost/performance (\$/bit-second) must improve by two orders of magnitude (10x the devices at 10x the data rate)
 - Even relaxing this constraint somewhat (10x higher system cost) requires one order of magnitude better cost/performance
 - If integrated into a switch/NIC, this represents the additional cost for the additional functionality
 - If as a standalone device, this is the total cost
- This will require substantially different system design
 - Can't use heavyweight algorithms for worm detection
 - Will probably require hardware implementations
 - Probably will benefit from integrating with switches/routers

Hard-LANs and False Positives

- If we measure false-positive rate in positives/bit, Hard-LAN devices will need at least 2 orders of magnitude improvement over conventional NIDS
 - And NIDS is currently too noisy for many automatic responses
 - Thus the reduction in false-positive-rate may need to be greater
- Fortunately, the LAN is better constrained
 - Should help reduce the noise for algorithms
 - E.G. can ban rendezvousless P2P systems, use Windows Domains instead of Workgroups
 - Both P2P without rendezvous points and Windows Workgroups include scanning activity, which creates a false positive on scan-detectors
 - But the enterprise LAN hasn't been as well studied
 - Especially within the academic context

Conclusion

- Worms bypass our current defensive postures
 - Firewalls and NIDS are easily circumvented
 - End-host based systems are brittle
 - Significant limitation for future worm defenses
- Thus we believe that much of the defense will need to be LAN centric:
 - Either in the switches, NICs, or as separate devices in the LAN
 - Hard-LANs designed to detect and respond to worm attacks
 - Will require substantial improvements over the current state of the art
 - 1-2 orders of magnitude better cost/performance
 - 2+ orders of magnitude lower false-positives
- But constraints may be achievable
 - Example algorithm for scan suppression:
2 memory locations accessed per packet
 - Details are in paper and at Usenix Security

Backup: An Example Hard-LANs Algorithm

- Scanning worms: search for "random" addresses and attempt to infect
 - The most common form of worm to-date
- Scanning-Worm Containment: Limit a scanning worm's spread within the institution
 - Goal is to limit a worm's infection to a small part of the network
 - Assumes that the general perimeter is imperfect, but still mostly good
 - Only a few penetrations will occur
- Operates by detecting and blocking scanning
 - Necessary behavior for a scanning worm
 - Can report this activity to other devices/systems
- Just one example of how algorithms are modified to deal with Hard-LAN style requirements
 - More details in [WSP04] at Usenix Security

Backup: Goals for Scan Detection

- Want an algorithm which is suitable for low-cost hardware implementation
 - Stand-alone device or integrated into a switch ASIC
 - 8 MB working set, 2 locations accessed per packet
- Began with a known good algorithm, TRW (Jung et al)
 - Assumes that scans fail > 50%, normal connections fail <50%
 - Every IP has a counter. For every connection attempt by that IP, consult a "connection oracle"
 - If connection succeeds, count = count - 1
 - If connection fails, count = count + 1
 - If count = 5, system is a scanner
 - Based on a mathematical analysis
- Limitations:
 - Connection oracle is offline: it needs to wait to see if the connection succeeds or fails
 - Possibly tracking all IPs
 - Possibly tracking all pending and active connections

Backup: Modifying Scan Detection For Hard-LANs application

- Replace the connection oracle with "guilty until proven innocent"
 - On a connection attempt, increase the count by 1
 - When connection succeeds, reduce the count by 2
 - Enables on-line operation
- Track connections with an approximate table
 - $H(\text{OutsideIP}, \text{InsideIP}, \text{OutsidePort}, \text{InsidePort}) \rightarrow \text{index}$
 - Entry just marks directionality and idleness
 - Idle connections removed after a period of time
 - Creates a false-negative due to aliasing/combining
 - Requires accessing only a single memory location
- Track external IPs with an approximate cache
 - $E(\text{OutsideIP}) \rightarrow \text{Index/Tag}$
 - Use associative cache to find entry
 - Evict most-negative on conflicts
 - Again, only requires a single access
- Thus algorithm can meet our goals
 - <8 MB working set in 2 tables, only 2 read & writes per packet
 - Can use a single DRAM for a bidirectional Gb implementation