Using Packet Symmetry to Curtail Malicious Traffic

Jon Crowcroft
Christian Kreibich(mostly), Andrew Warfield, Steven Hand, Ian Pratt
The Computer Laboratory
University of Cambridge
http://www.cl.cam.ac.uk/~jac22
Any questions?

- To start with:-)
- Btw, we have other work (Manuel Costa/Microsoft) on Worm Containment, which I can talk about if you like)
- But this is most relevant to DOS...->
A word from our sponsor

- Communications Research Network
  - CMI funded (UK/US, +BT/BP et al)
  - Network of industry+academics
    - BT, Cisco, Juniper, Nokia, etc
    - UCL, Cambridge, Oxford, MIT
- Working Groups
  - Core Edge+Broadband, Interprovider Routing+QoS,
  - Security, Denial-of-Service
  - Open Spectrum, Photonics
What’s Malicious

- Anything that’s not typical
  - Typically, traffic dynamics can be observed
- What is a very simple, immediate characteristic that can be used:
  - Implicitly, to allow or deny, or
  - limit atypical behaviour at the ingress to the net
- Before its “too late”
  - reactive response is far too slow for DDoS attacks
Smoke and Mirrors

- Most flows are roughly symmetric at the packet level
  - Whenever a packet is sent, a packet is received within some reasonable interval (round trip time)
    - This can be measured (and enforced) at the edge router inexpensively
  - It is remarkably robust
    - And surprisingly universal!
- Nicely orthogonal to simple blocking based on default allow/deny at ISP boundaries
  - It doesn't operate on a per-flow level
Ingress versus Egress

- Firewalls ok to stop bad stuff at ingress to sink.
- Too late for DoS - need egress defense near source
- server (e.g Xen) farm v. ISP deployment considerations
Asymmetry metric

- $S = \ln \frac{(tx+1)}{(rx+1)}$
  - Seems suitable since it is negative for $rx>tx$,
  - 0 for $tx==rx$
  - And positive for $tx > rx$

- Note, $tx$ and $rx$ are packet count **not** byte counts
- Need to be measured near transmitter
  - otherwise path asymmetry problem or address translation or spoofing problems

- Action is to delay, then drop
Prototypical Implementation

- Linux netfilter/iptables, Libipq
- Choose threshold $S = 2$ (asymmetry of 8 times)
  - If $S > 2$, delay $n$th subsequent packet by $2^n$ ms
  - If $S$ goes below 2, decay delay back to zero.

- Let’s see some data
Delay imposed on asymmetric flows

Illustration of asymmetry-based rate limiting
- Asymmetry
- Delay

Unacknowledged packets transmitted

Asymmetry value

Delay (ms)

0

0

256

512
A UDP Flood

Unacknowledged UDP flood of 1KB packets (no limiting)

Packet Rate (K packets/s)

Asymmetry

Elapsed Time (s)
A UDP Flood stemmed
A large, but normal (well behaved) TCP Flow

scp-based transfer of a 128MB file

Packet Rate (K packets/s)

Asymmetry

Elapsed Time (s)
Host based symmetry

Cumulative host packet symmetry, 170K hosts, varying windows

Percentage of hosts

$log\left(\frac{t+1}{n+1}\right)$

Red: 24h window
Green: 1h window
Blue: 1min window
Pale blue: 1s window
Host pair based symmetry

Cumulative host pair symmetry for responding hosts, 360K pairs, 60s window
Flow based symmetry

Cumulative maximum TCP flow asymmetry, 55,000 flows, 60s window

Percentage of flows

$\log(t_{s+1}/t_{c+1})$

- Red: Ignoring no packets
- Green: Ignoring first 10 packets
- Blue: Ignoring first 100 packets
UDP flow based symmetry
Evasive Manoeuvres

- Source address spoofing
  - Bad guy can masquerade as a good site
  - But they can’t get traffic back so won’t work
  - But they might cause good guy to get throttled...so:

- Randomization of IP ID
  - Bad guy cannot tell what IP ID from good guy can do
  - Policer/limiter can check the ID before throttling

- TTL Estimation
  - Bad Guy doesn’t know what TTL is from good guy
  - Policer can check TTL is “right” before throttling
Deployment considerations

- Part of Xen toolkit (virtualised device stuff)
- Behoves us to do this as Xen is likely to be deployed in high capacity (dangerous source potential) sites
- Could put in NIC
- Michael Dales (Intel) designed it into his optical switch port controller (Xilinx)
- Also proposed in ADSL DSLAM equipment (simple as part of ATM mux level police/symmetry enforcement in broadband access contention control).
Practical Protocol Considerations

- TCP acks every other packet 99% of the time
- UDP use:
  - DNS, SNMP - request/response
  - RTP/UDP - RTCP reports about 1/6th of RTP
- Counter examples
  - Syslog is only 1 we could find in BSD/Linux/OSX
  - Some Windows apps (DCOM use for Outlook:)
  - Almost all (100%) LAN only by definition:
  - Consequence of congestion control need in WAN?
Related work

- Other approaches require trace-back and/or push-back
  - Too expensive, too slow and too late
- Deal with symptom not cause!
  - more feasible for ISP as “bit-pipe provider" to deploy symmetry enforcement
  - than to filter traffic based on application-layer characteristics
- More fundamental architectural change
  - Mothy (hotnets 03?) - capability to send
  - Cheriton et al (to appear) - meta-capability
  - Handley/Greenhalgh (sigcomm 05) - asymmetry
Generalise?

- Should all protocols be mandated symmetric?
  - The “Well Tempered Internet” (Steven Hand’s piano player:)
  - Is this a design principle for feedback based systems?
  - Argue for both stability and for information theory reasons, hard to see otherwise...
  - Details (state/accuracy and asymmetry tradeoffs) TBD

- Acknowledgements to Mark Allman, Vern Paxson, Chema Gonzales, Juan Caballero, Michael Dales (200 lines of VHDL), Atanu Ghosh, Andrew Moore (traces)
Questions?

• Any?
  • Q1. Can you devise a symmetric attack? (Nick Mckeown&M. Andrews from Bell Labs)
    • A1. Yes, but hard for bad guy coordinate, so easy for ISP to detect
  • Q2. What about randomizing the initial slow down value to make it hard to for bad guy to probe for symmetry policers? (Stephen Farrell from TCD asked this one!)
    • A2. Cool!
  • Q3. Isn’t there a more general principle in this symmetry idea? (Ted Faber from ISI)
    • A3. Guess so...
Cumulative host pair symmetry for non-responding hosts, 6835K pairs, 60s window.
Cumulative host pair symmetry for non-responding hosts, 6835K pairs, 60s window.