High Performance Traffic Shaping for DDoS Mitigation

Markus Goldstein, Matthias Reif, Armin Stahl, Thomas Breuel

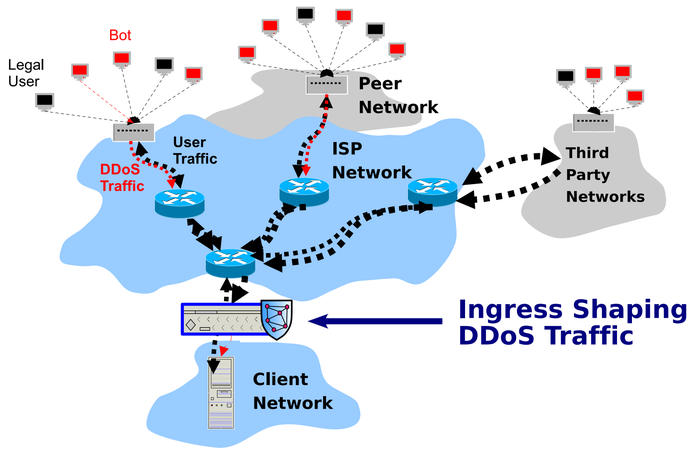
Technical University of Kaiserslautern, Germany
German Research Center for Artificial Intelligence (DFKI), Kaiserslautern

Introduction

- DDoS flooding attacks (UDP, ICMP, SYN) easy identifiable
- TCP based protocol conformant DDoS attacks (e.g. HTTP-GET floods) have lower deviation compared to normal user traffic
  → more difficult to distinguish valid and illegal traffic
  → firewall DROP rules may cause blocking legal users

Idea

- Instead of blocking/accepting flows we shape their source IPs



- *Conditional Legitimate Probability (CLP)* [5] is the probability of a flow to be legal

$$CLP(p) = \frac{N_n \cdot P_n(A = a_p) \cdot P_n(B = b_n) \cdot \ldots}{N_m \cdot P_m(A = a_p) \cdot P_m(B = b_n) \cdot \ldots} \quad (1)$$

where N_n is the number of normal packets, N_m the number of measured packets (mixture of normal and attack traffic) and $P(A = a_p)$ the probability of a feature A to be a_p.

- CLP is calculated on the basis of previously observed traffic, e.g. histograms on source IP prefixes, packet sizes or server ports
- The higher the CLP, the higher the assigned bandwidth during a DDoS incident

Challenge

- During an DDoS attack, thousands of source IP addresses need to be shaped.
- On Linux systems, tc [2] can only handle few shaping rules
  → Need for a better performing algorithm for the DDoS scenario.

Traffic Shaping

- Easy and fast algorithm for high packet rates
- Specialized for DDoS mitigation (filter parameter only source IP)
- IP ranges are continuous intervals r over IP addresses $[r^{start}, r^{end}]$ with a defined bandwidth limit
- List of ranges can be sorted to perform binary search:

$$\forall r_i, r_j \in R, i < j : r_i^{end} < r_j^{start} \quad (2)$$

- Every arriving packet is accepted, queued or dropped, similar to *Random Early Detection (RED)* [4] in the PACKET_HANDLER function.
- A triggered function TIMER_HANDLER sends packets (with respect to the defined bandwidth) and calculates the used bandwidth
- Complexity for each incoming packet is $O(\log_2 n)$, where n is the number of IP ranges with bandwidth limits.
- Worst case complexity: every source IP address has a different target bandwidth $O(\log_2 2^{32}) = 32$ lookups for each incoming packet.

Future Work

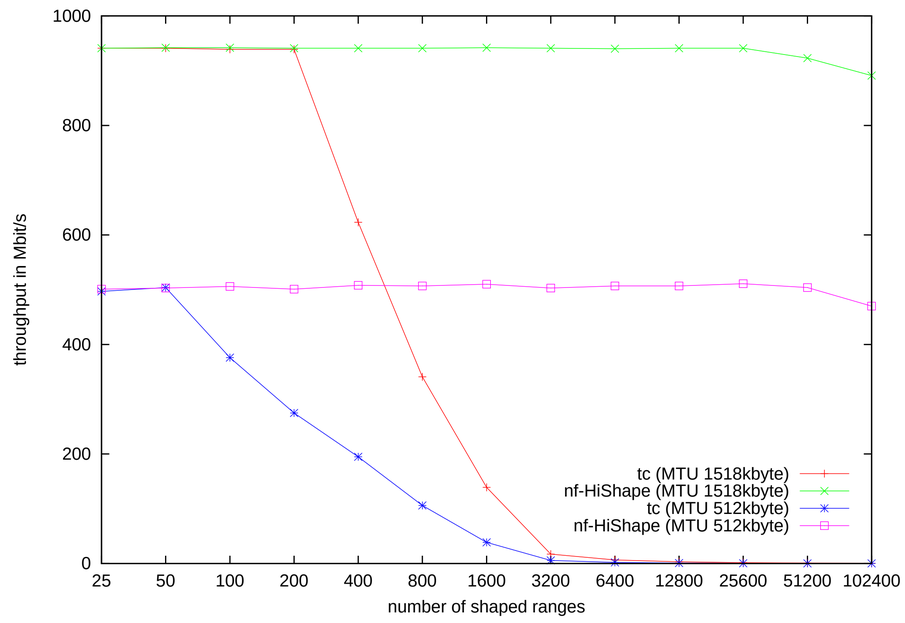
- Support more than one parameter for shaping
- Investigate FIS trees [3] with $O(\log_2 \log_2 N)$ lookup complexity

Shaping Algorithm

```
1:  function PACKET_HANDLER(Packet p)
2:      r ← range including. p.source_IP using binary search
3:      if r not found then
4:          accept(p) and return
5:      q ← queue of r
6:      if not q.empty or r.sent+p.size > r.limit then
7:          if q.size < q.max_size then
8:              q.push(p)
9:              steel(p)
10:         else drop(p)
11:     else
12:         r.sent + = p.size
13:         accept(p)
14: function TIMER_HANDLER
15:     for all ranges r do
16:         r.sent ← 0; finished ← false
17:         q ← queue of r
18:         while not q.empty and not finished do
19:             p ← q.front()
20:             if r.sent + p.size < r.limit then
21:                 send(p)
22:                 q.pop()
23:                 r.sent += p.size
24:             else finished ← true
```

Evaluation

- Measuring throughput of a legal (not shaped) user on a 1GBit/s link depending on the number of shaped IP ranges.
- *tc* throughput decreases at 400 shaped ranges, not enough to mitigate DDoS attacks
- *nf-HiShape* still performant at 100,000 defined IP ranges



Open Source Software

The presented algorithm is implemented as a Linux kernel module called *nf-HiShape* [1].
It is available under GPL license at **http://code.google.com/p/nf-hishape/**.
It ships with a userland tool which syntax is similar to iptables, e.g.:

```
nf-hishape -i eth0 -f 192.168.0.0 -t 192.168.255.255 -l 5.0
```

sets a limit for 192.168.0.0/16 to 5.0 kB/s and

```
nf-hishape -i eth0 -L
```

lists all shaping ranges for the given interface. The userland tool also reads in ASCII files with bandwidth limits for faster integration with DDoS mitigation systems.

References

[1] nf-hishape. http://code.google.com/p/nf-hishape/.

[2] W. Almesberger. Linux Network Traffic Control - Implementation Overview. In *5th Annual Linux Expo*, pages 153-164, 1999.

[3] A. Feldmann and S. Muthukrishnan. Tradeoffs for packet classification. In *INFOCOM*, pages 1193-1202, 2000.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1:397-413, 1993.

[5] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao. Packetscore: A statistics-based packet filtering scheme against distributed denial-of-service attacks *IEEE Transactions on Dependable and Secure Computing*, 03(2):141-155, 2006.

[6] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security). *Prentice Hall PTR*, Upper Saddle River, NJ, USA, 2004.