Mayday:
Distributed DoS Filtering

David Andersen

MIT Laboratory for Computer Science

March 2003

`http://nms.lcs.mit.edu/ron/`

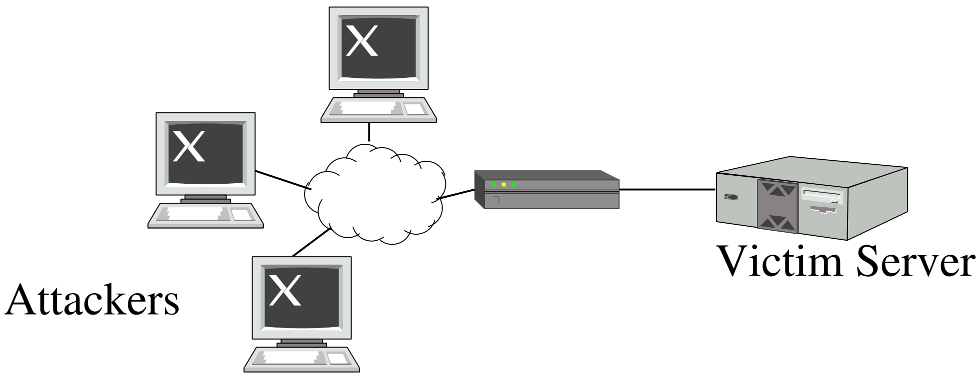
Proactive Defense against DoS

- Many systems *trace* DoS attacks
- Some *react* to DoS attacks
- A few *prevent*, but
  - ✘ Require near-global deployment, or
  - ✘ Don't protect outside of your own network

✔ Mayday:
  - incrementally deployable
  - proactive defense

Flooding Attacks

- Overload servers (not “ping of death”)
- Probably have lots of attack machines...
- ... and can spoof IP addresses
- We’ll discuss more powerful attackers later

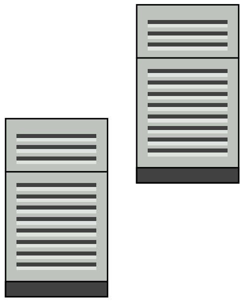


Overlay Nodes and Filtering Routers

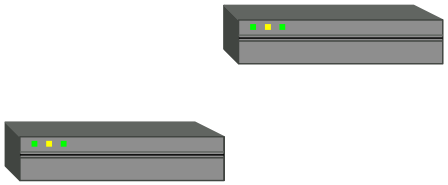
Borrow an idea from SOS

(Secure Overlay Services, [Sigcomm 2002]):

Use overlay nodes and normal routers to protect servers.

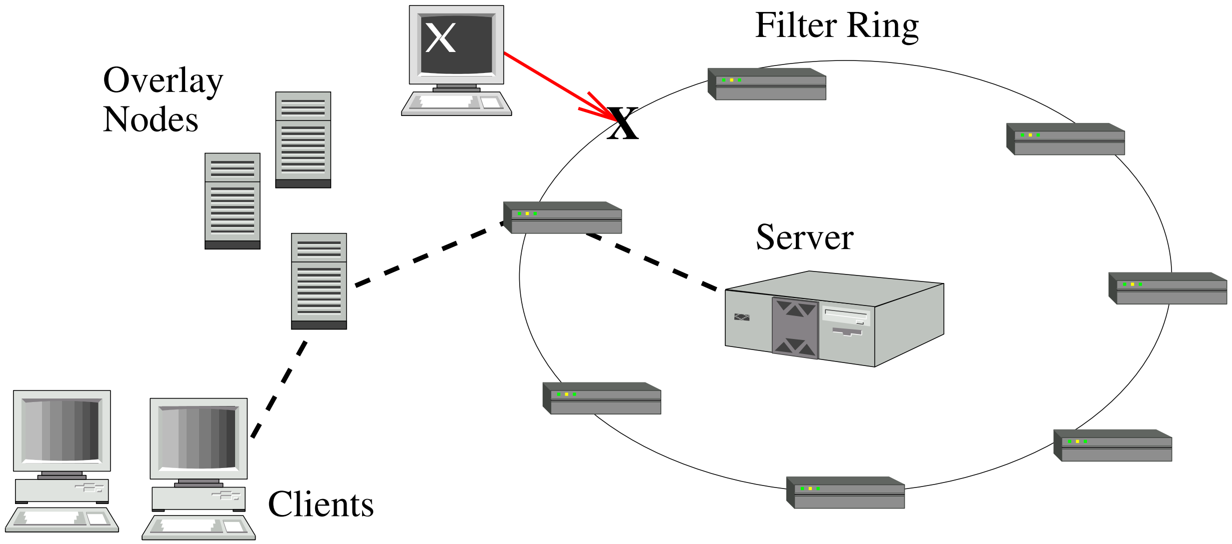


Overlay Nodes
(10s to 1000s)



Routers
(10s)

Overlay Nodes and Filtering Routers



- Routers allow only “good” traffic in
- Overlay nodes are “good” traffic
  - verify that clients allowed to use service

→ How?

Roadmap

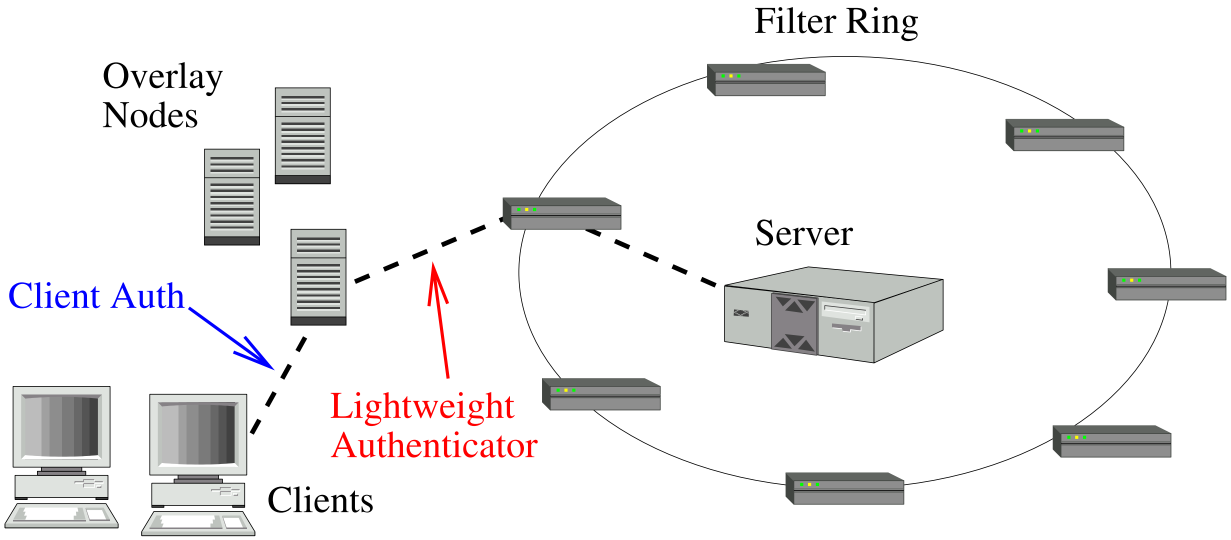
- Expand and understand design space
- Evaluate the architecture
  - Earlier work: DoSing nodes futile
  - Our work: More sophisticated attacks
- Attacks necessitate expanded design space...

Making it practical

- Effective filtering must be near “core”
- Set of allowed clients dynamic or large
- Core routers can’t do heavy-duty filtering
- Let’s use existing router capabilities

✘ IPsec to the filter routers is a no-go.

Architecture



- Clients authenticate to overlay nodes
  (Can be heavy, not our concern)

- Overlay nodes authenticate to filter ring
  → Lightweight Authenticator

Lightweight Authenticators

- Source Address
  - ✔ Well understood, good with no spoofing
  - ✘ Limited # of correspondant nodes
  - ✘ Updated by filter changes

Lightweight Authenticators

- Source Address
  - ✔ Well understood, good with no spoofing
  - ✘ Limited # of correspondant nodes
  - ✘ Updated by filter changes
- Server Destination Port
  - ✔ Larger key space (65,000)
  - – Many correspondant nodes
  - ✘ Updated by filter changes

Lightweight Authenticators 2

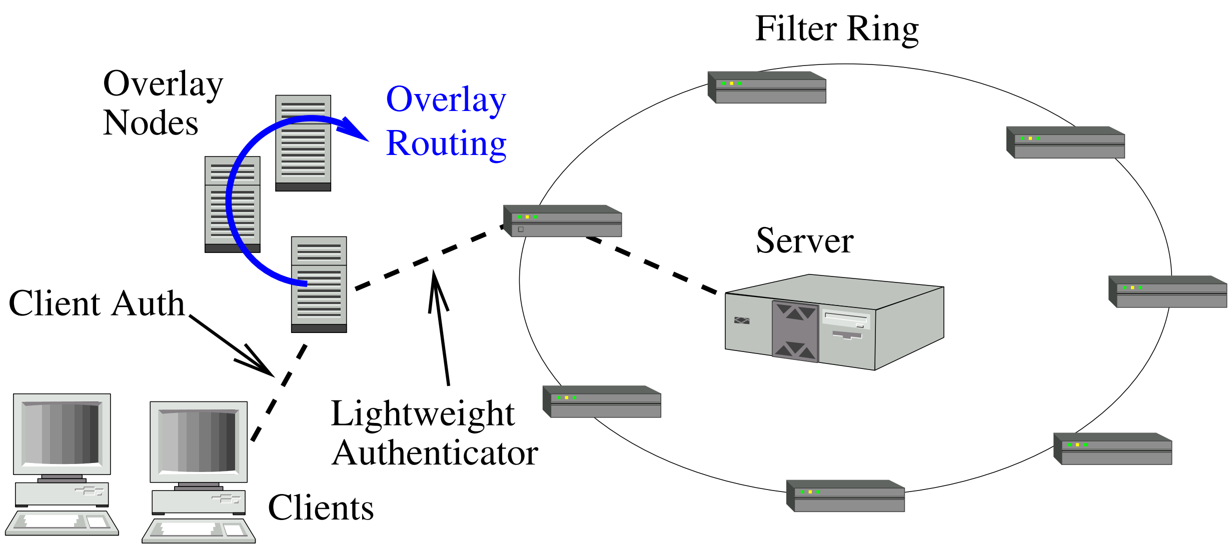
- Server Destination *Address*
  - ✘ Small key space
  - ✘ Changes IP address
  - ✔ Updated via fast routing protocols

Lightweight Authenticators 2

- Server Destination *Address*
  - ✘ Small key space
  - ✘ Changes IP address
  - ✔ Updated via fast routing protocols
- Other header fields
  - ✔ Adds to key space
  - ✘ Not all routers support

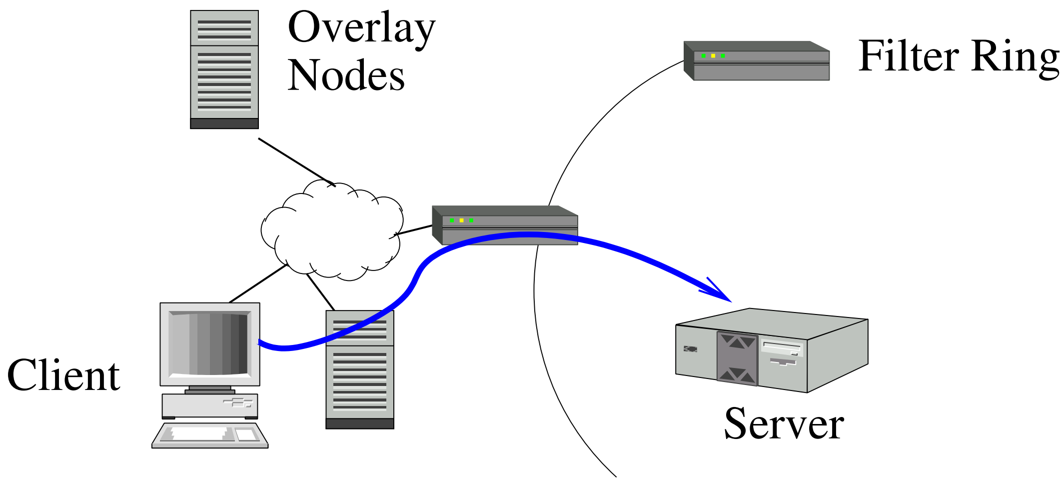
Overlay Routing Improves Security

- Fewer nodes have direct access to server



Choice of routing depends on authenticators, paranoia.

Overlay Routing: Proximity



- ✔ Like Akamai, great performance
- ✘ All nodes possess authenticator
- ✘ Can't rely on source address auth

Overlay Routing

- Proximity Routing
- Singly-Indirect Routing
  - Ingress node passes to egress node
  - ✔ Fewer nodes know authenticator (except for source address)

Overlay Routing

- Proximity Routing
- Singly-Indirect Routing
- Doubly-Indirect Routing
  - Only a few nodes know the egress node
  - ✔ Keeps source auth secret
  - ✘ Overhead grows...

Overlay Routing

- Proximity Routing
- Singly-Indirect Routing
- Doubly-Indirect Routing
- Random or Mix Routing
  - Route through many overlay nodes
  - ✔ Resistant to node compromises
  - ✘ Overhead grows more...

Choose protection vs. Overhead

What authenticator / routing combinations?

- Performance: Proximity non-source
  - vulnerable to eavesdroppers
- Eavesdropping: Singly-indirect non-source
  - Random eavesdroppers don't know secret
  - Equivalent security to SOS, fewer hops

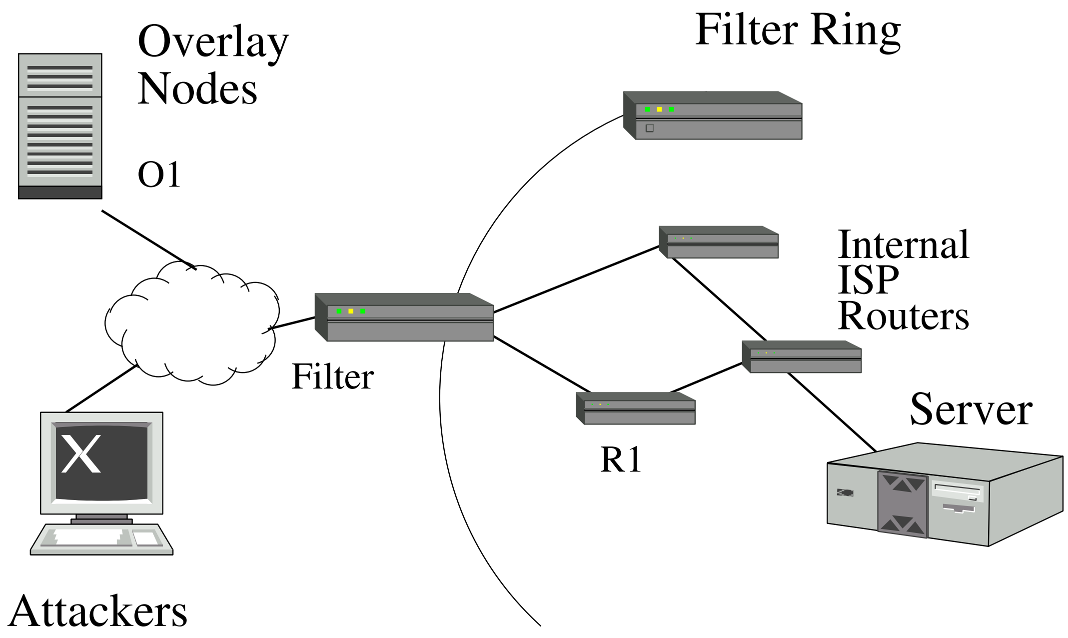
Choose protection vs. Overhead

What authenticator / routing combinations?

- Agility: Singly-indirect destination
  - Routing updates can change filters
  - Resists adaptive attacks (discussed next)
- Maximum Security: Mix routing
  - Like Freenet
  - Resists some overlay node compromises

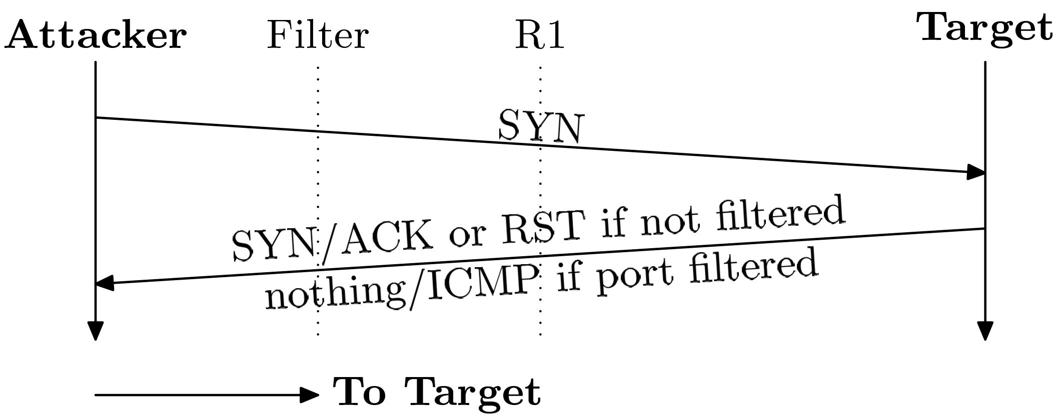
Using more authenticators boosts the key space

Attacks and Defenses



- Basic flooding resistance shown already
- Real networks have third parties, traffic can be sniffed, etc.

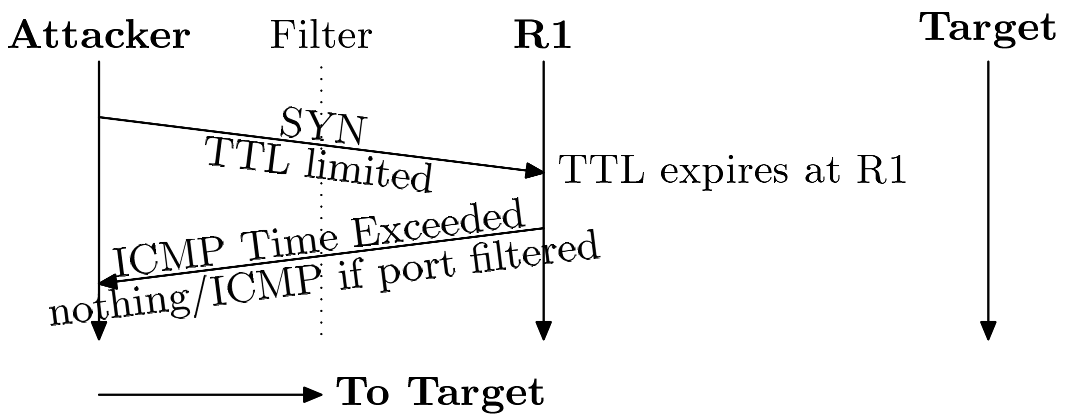
Probing: Basic



- ✘ About 30 seconds to find destination port
- ✔ Secondary Key -

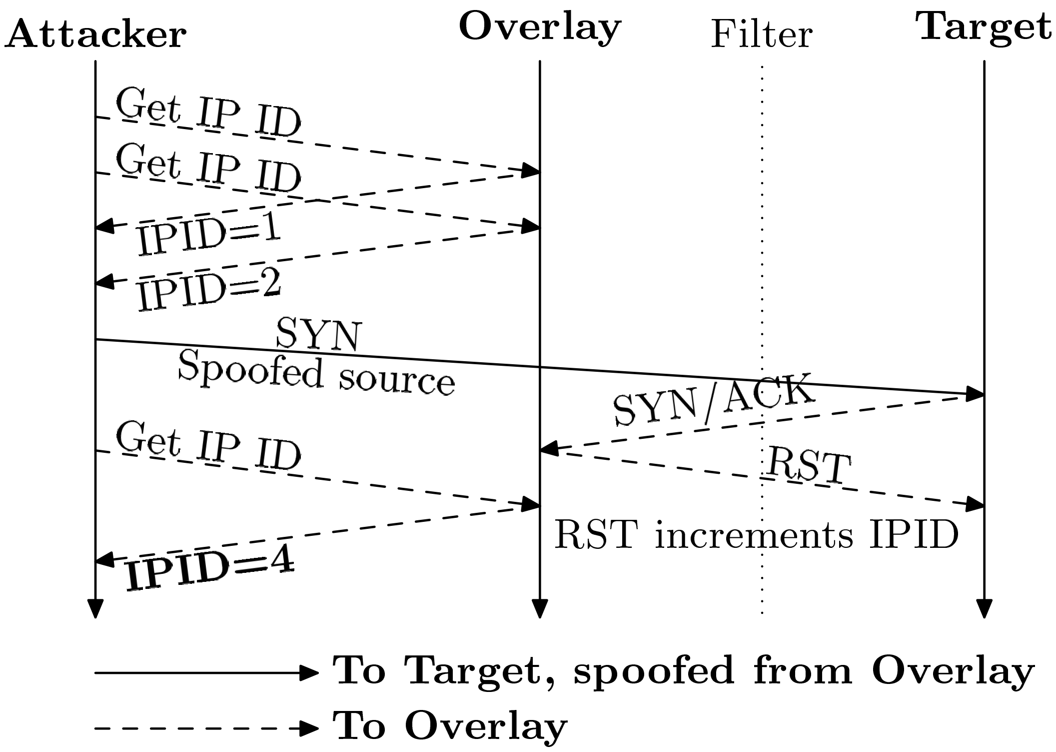
  server only responds to good requests.

Probing: Secondary Key



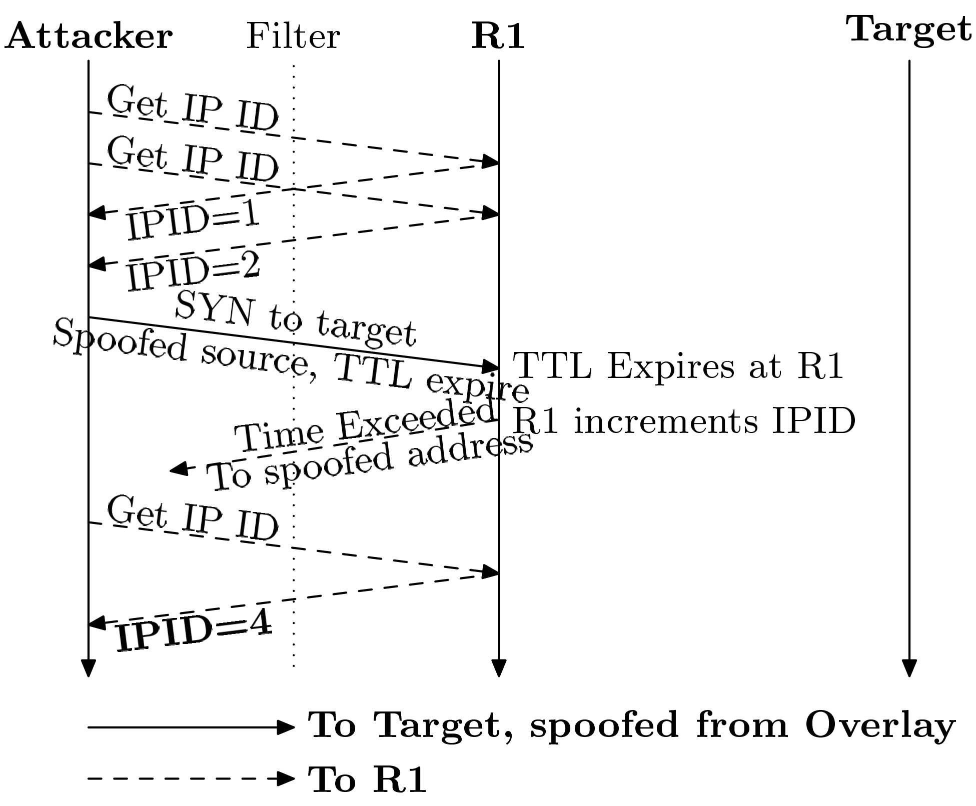
- ✘ Use Firewalking against intermediate routers
- ✘ ... about five minutes to port scan.
- ✔ Fix intermediate routers (ick)
- ✔ Use source address authentication

Probing: Secondary Key + Source



- ✘ Use Idlescan via overlay nodes
- ✔ Fix overlay nodes

Probing: Secondary Key + Source



- ✘ Next-hop scan via routers
- Fix everything...

Further Attacks

- Timing Attacks
  determine egress node
- Adaptive Flooding
  smarter flooding, detect slowdown
- Request floods, compromised nodes...
- Shameless plug: All discussed in paper

How big are attacks?

(most data from Savage et al.)

- 30% of attacks $\geq$ 1000 pps
- 5 % $\geq$ 10,000 pps

Large keyspace resists attacks

At 1000 pps, how long can we resist attack?

- ✘ Port-scan dest port: 5 minutes
- ✘ Locate egress node: 50 seconds
- ✔ Find both: 4 days
- Agility: update when discovered

Is any of this practical?

We think so!

- ✔ CDNs like Akamai obvious deployment path (Akamai offers “mayday-lite”)
- ✔ New core routers can filter at line-speed
- Useful in a service-provider context
  - Amortize costs, load spikes
  - Not everyone attacked at once.

Conclusions

- ✔ Practical, proactive DoS resistance
- ✔ Flexible choices of overhead vs. protection
- ✔ Better understanding of attacks
  (next-hop attack and adaptive flooding novel)
- Only the first line of defense!
  Security starts at home.