The Spoofer Project

Inferring the Extent of Source Address Filtering on the Internet

Rob Beverly and Steve Bauer
{rbeverly,bauer}@mit.edu

The Spoofer Project

Goal:

- Quantify the extent and nature of source address filtering on the Internet

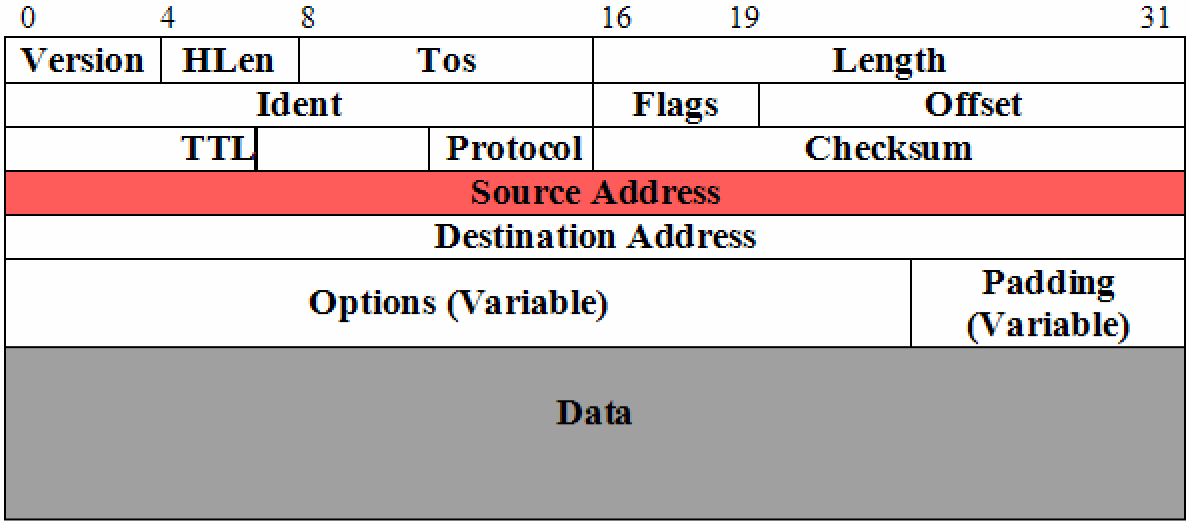
Key results:

- ~23% of observed netblocks corresponding to ~24% of observed ASes allow some from of spoofing
- Filtering is frequently applied inconsistently allowing spoofing of parts of the address space
- Filtering policies corresponds reasonably well to netblocks announced in BGP
- No discernable geographic pattern in address filtering policies

Motivation and background

What are spoofed packets?

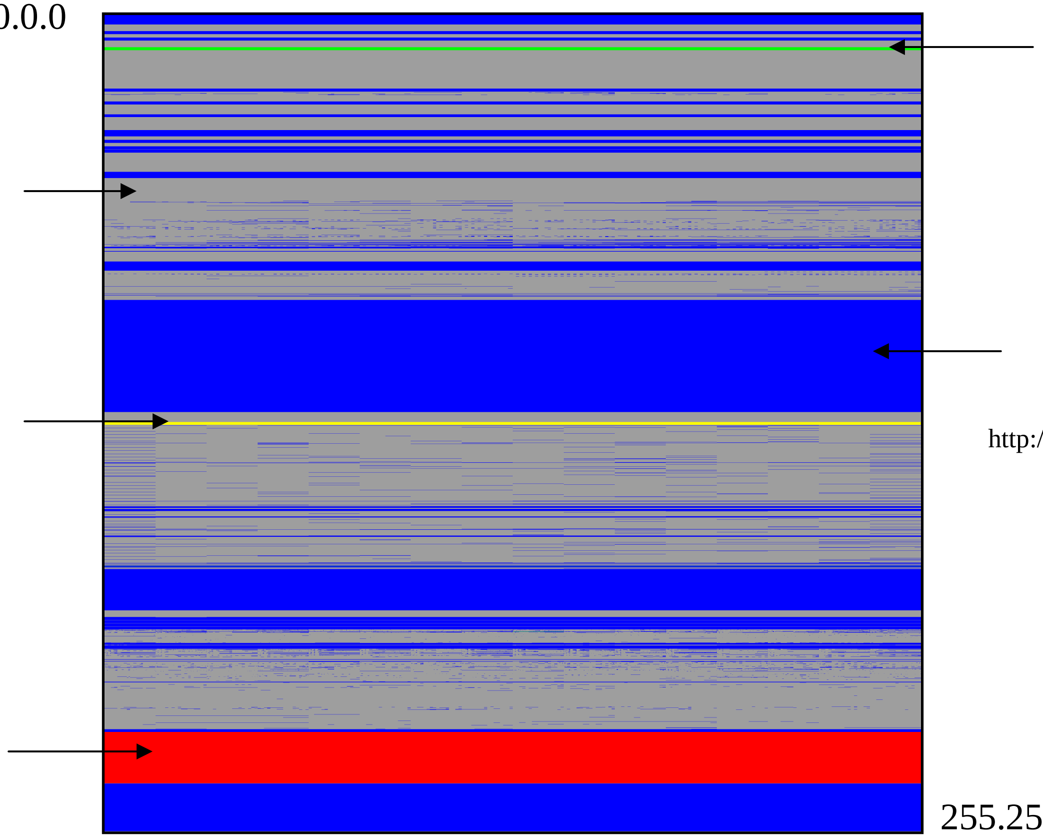
- Attackers/compromised-hosts forge or “spoof” source address of an IP packet



0	4	8	16	19	31
Version	HLen	Tos	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
Source Address					
Destination Address					
Options (Variable)				Padding (Variable)	
Data					

What type of addresses are spoofed?

IPv4 Address Space



0.0.0.0

Private Intranet

10.0.0.0/8
172.16.0.0/12
192.168.0.0/16

Valid

Unallocated

June 29, 2005
http://www.completewhois.com/bogons/

Loopback

127.0.0.0/8

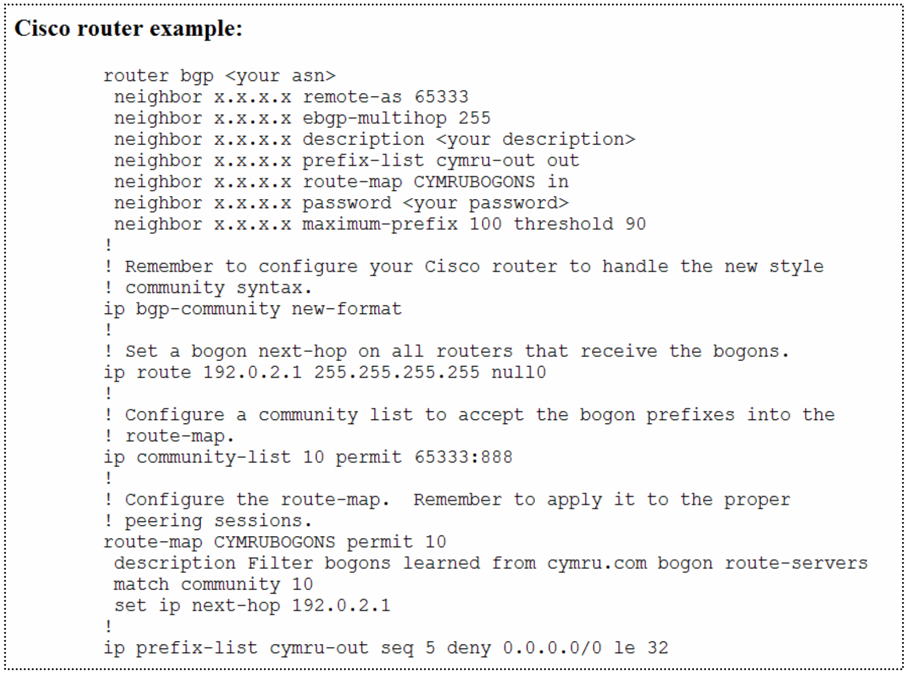
Multicast

224.0.0.0/4

255.255.255.255

How are bogons filtered?

- Bogon list sources:
  - http://www.cymru.com/Bogons/
  - http://www.completewhois.com/bogons/
- Ingress or egress filters on a router
- Need updating (ideally automatically) as assignments change
- Not always 100% accurate



Cisco router example:

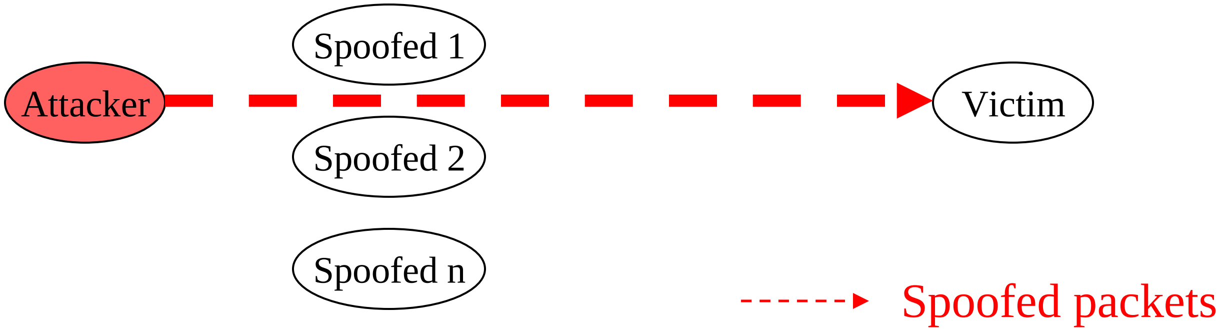
```
router bgp <your asn>
 neighbor x.x.x.x remote-as 65333
 neighbor x.x.x.x ebgp-multihop 255
 neighbor x.x.x.x description <your description>
 neighbor x.x.x.x prefix-list cymru-out out
 neighbor x.x.x.x route-map CYMRUBOGONS in
 neighbor x.x.x.x password <your password>
 neighbor x.x.x.x maximum-prefix 100 threshold 90
!
! Remember to configure your Cisco router to handle the new style
! community syntax.
ip bgp-community new-format
!
! Set a bogon next-hop on all routers that receive the bogons.
ip route 192.0.2.1 255.255.255.255 null0
!
! Configure a community list to accept the bogon prefixes into the
! route-map.
ip community-list 10 permit 65333:888
!
! Configure the route-map.  Remember to apply it to the proper
! peering sessions.
route-map CYMRUBOGONS permit 10
 description Filter bogons learned from cymru.com bogon route-servers
 match community 10
 set ip next-hop 192.0.2.1
!
ip prefix-list cymru-out seq 5 deny 0.0.0.0/0 le 32
```

Does spoofing *matter* in 2005?

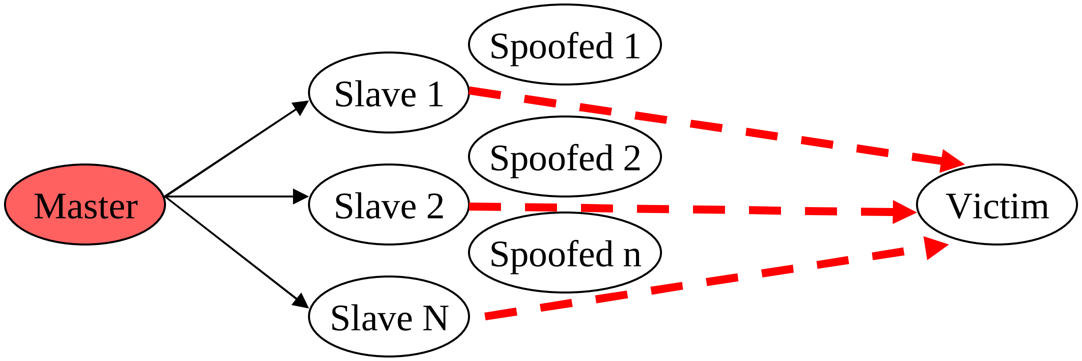
- All ISP filter (right?)
  - – RFC2827, uRPF
- Zombie farms
  - – Spoofing provides little additional anonymity for actual attacker
- Prevalence of NATs
  - – headers rewritten anyway so spoofing useless

Indications that spoofing is employed in current attacks

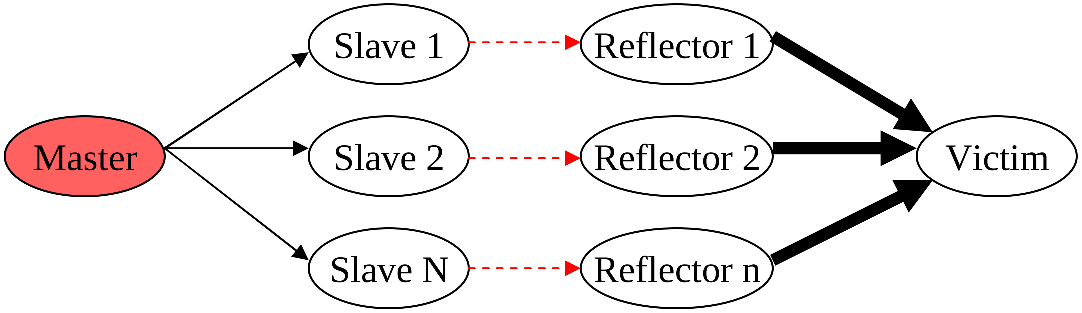
- Backscatter [Moore01][Pang04] shows ***continued, strong spoofing activity***
- In Jan 2005 during one DDoS attack 12% of the source addresses were bogons [Dietrich05]
- High-profile spoofing-based DDoS attacks in 2000-2004:
  - Yahoo, Ebay, E*trade
  - Shaft, TFN, trinoo, Stacheldraht, RingZero
  - Protx online payment site, Nov 2004



DoS attack with spoofing



Distributed DoS attack with spoofing



Distributed DoS attack with reflectors

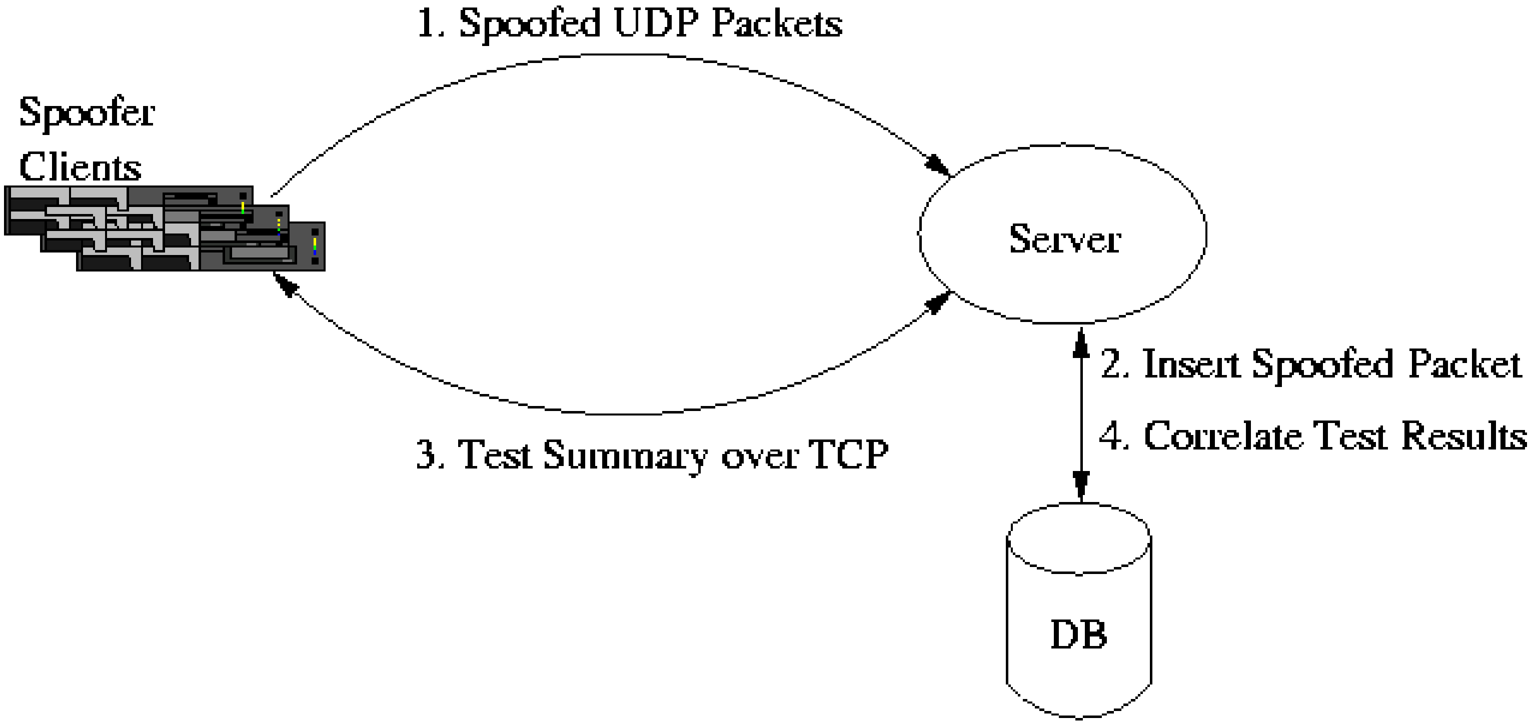
Prediction: spoofing increasingly a problem in the future

- **Spoofed traffic complicates a defenders job**
- Adaptive programs that make use of all local host capabilities to amplify their attacks
- Consider a 10,000 node zombie DDoS
  - Today (worst case scenario): if non-spoofing zombies are widely distributed, a network operator must defend against attack packets from 5% of routeable netblocks.
  - Future: if 25% of zombies capable of spoofing significant volume of the traffic could appear to come any part of the IPv4 address space

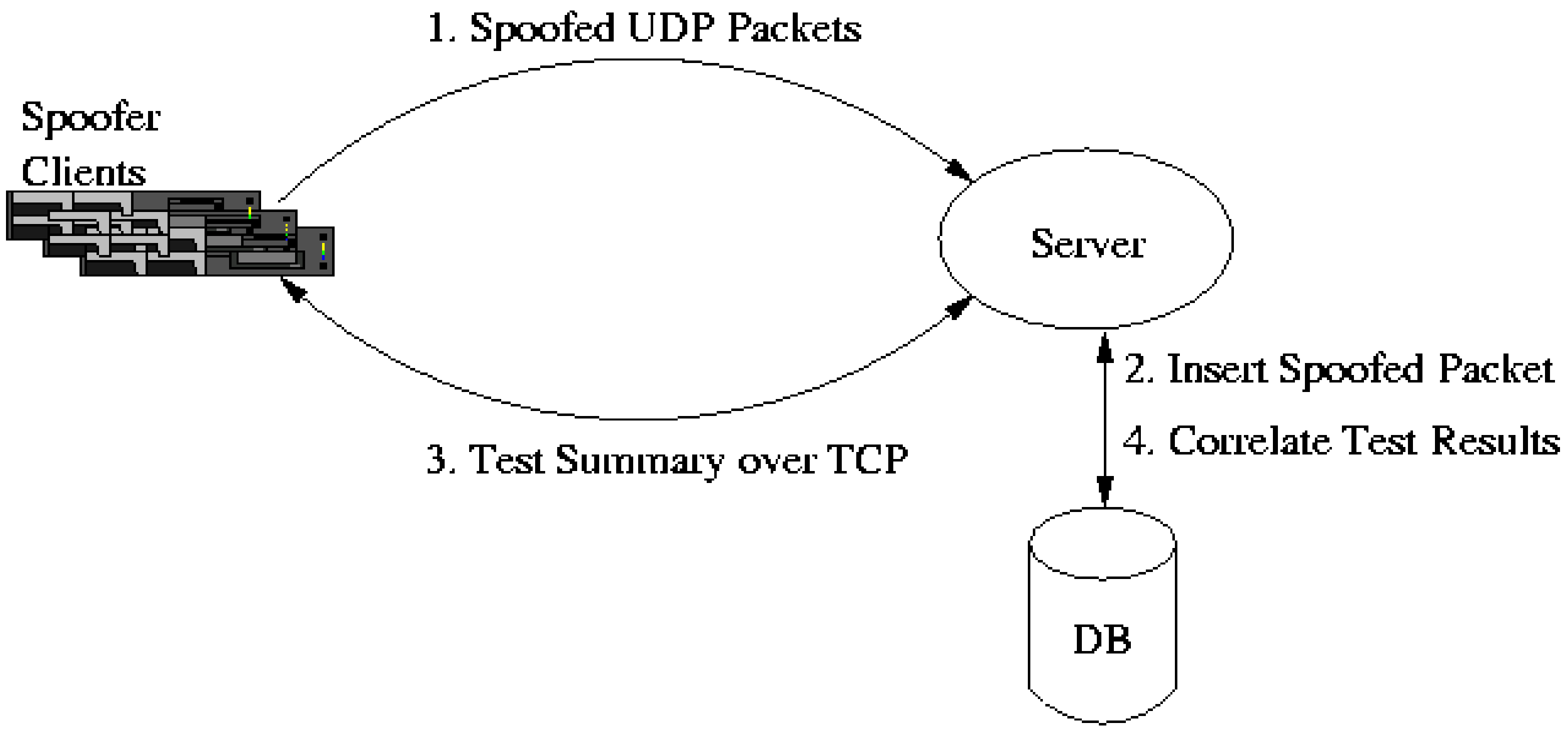
Spoofer Project: Collection and analysis methodology

Collection methodology

- **Objective: collect reports of the spoofing capabilities from as many locations on the network as possible**
- Spoofing packets requires administrator privileges
- No way to induce spoofed packets on remote machines
  - need willing participants, unavoidably introducing a *potential* bias
- Clients run a “spoofer” test program generating a report from their network locations
- Availability advertised on various mailing lists



1. Spoofer clients attempt to send a series of spoofed UDP packets to our test collection server
   - Five of each type with random inter-packet delay
   - UDP destination port 53 (normally DNS) to avoid secondary filtering effects
   - Payload includes unique 14 byte identifier
2. If received, server stores packets in database



3. Test summary

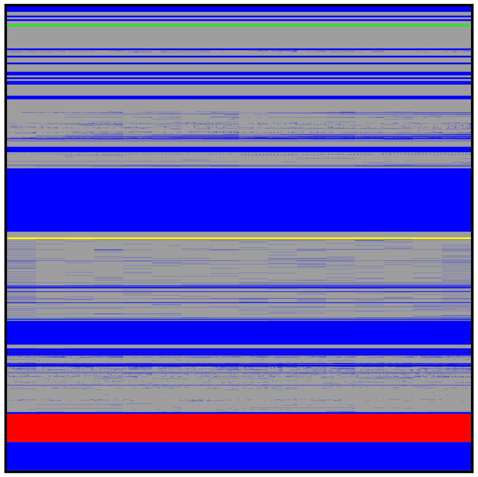
- Spoofer client does a traceroute to server
- Spoofer client sends a report of spoofed packets to server via TCP
- TCP destination port 80 used to avoid secondary filtering effects

Spoofed packets

- Chosen to infer specific filtering policies

Spoofed Source	**Description**
1.2.3.4	Unallocated
6.1.2.3	Valid (In BGP table)
172.16.1.100	RFC1918 Private address
Client IP ⊕ (2^N) for 0<N<24	Neighbor Spoof

IPv4 Address Space



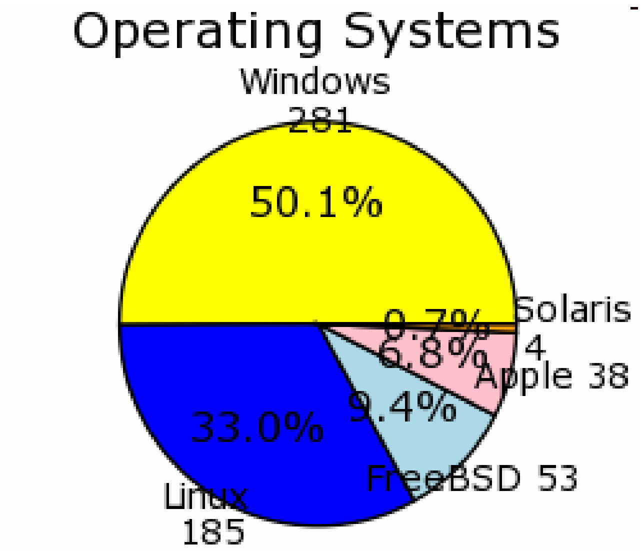
Example client run

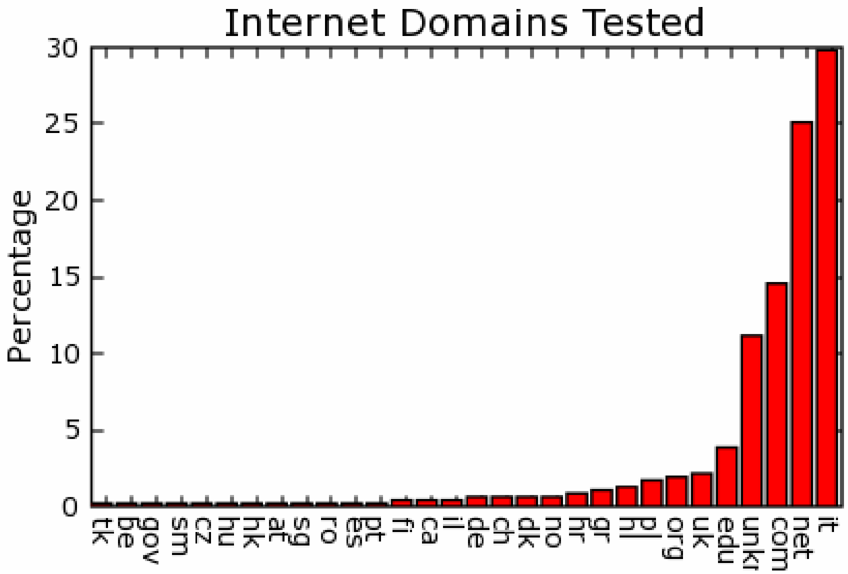
```
[root@coco spoofer]# ./spoofer
>> Spoofing Tester v0.2
>> Source 5 spoofed packets (IP: 1.2.3.4) (Seq: g8cb4gc6ojezw1)...
>> Source 5 spoofed packets (IP: 172.16.1.100) (Seq: 09kamtjjugxwvy)...
>> Source 5 spoofed packets (IP: 6.1.2.3) (Seq: 0dzpw2obc80ff3)...
>>
>> Checking spoofing result...
>> Server response: HOWDY 5am11w18zzc86g
>> Server response: COOL 3
>> Server response: FOUND g8cb4gc6ojezw1
>> Server response: FOUND 09kamtjjugxwvy
>> Server response: FOUND 0dzpw2obc80ff3
>> Running Trace (please wait): /usr/sbin/traceroute -n 18.26.0.235
traceroute to 18.26.0.235 (18.26.0.235), 30 hops max, 38 byte packets
>> Server response: SEND-TRACE LINUX
>> Server response: BYE 5am11w18zzc86g

Test Complete.
Your test results:
  http://momo.lcs.mit.edu/spoofer/report.php?sessionkey=5am11w18zzc86g
```

Analysis and results

Client population

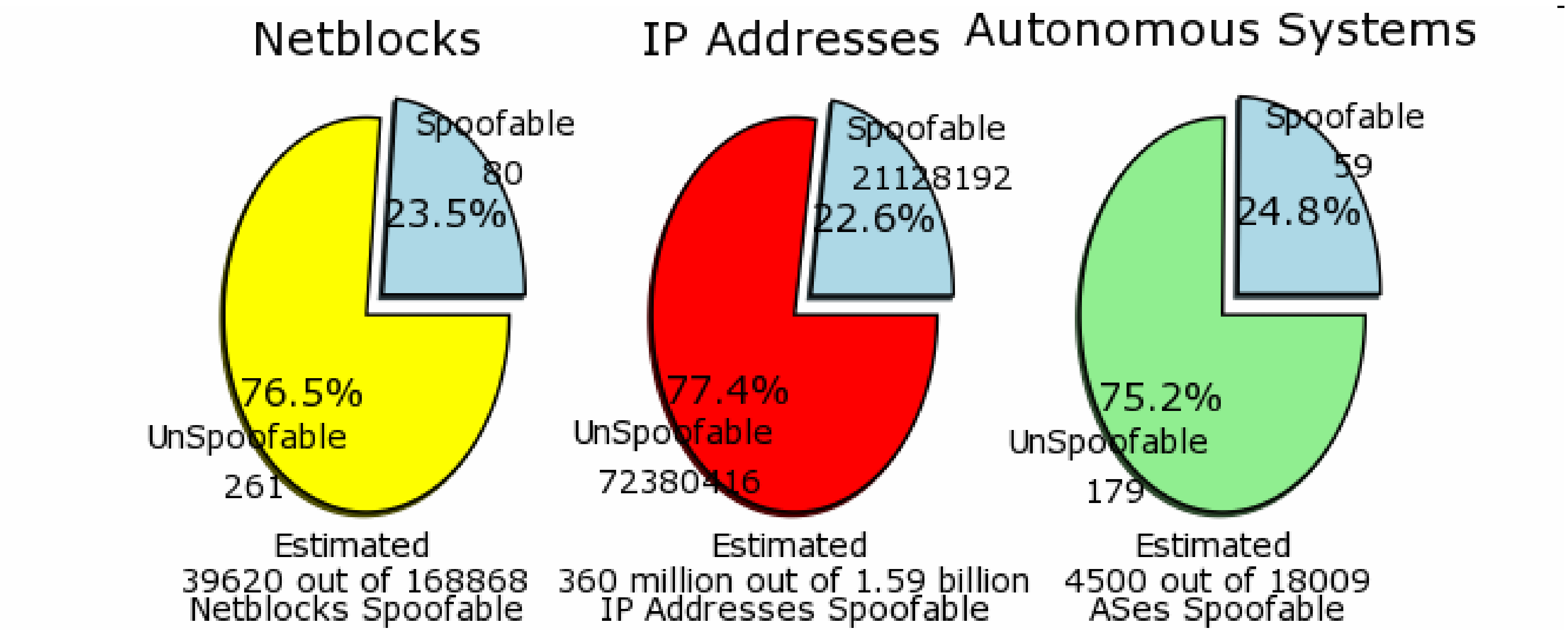
- From March 2005 to present:
  - 688 client reports generated
  - 544 unique client reports
  - No network abuse complaints reported from users or received by us





Spoofing failures for reasons not related to ISP policies

- Non-ISP related spoofing failures 326 client reports
  - Blocked by Windows XP SP2: 155
  - Hosts Behind NATs: 126
  - Otherwise blocked by operating system: 20
- **We exclude these from our analysis**
  - because they do not definitively provide any indication of the capability of other hosts in the same netblock to spoof



- **Spoofable:** spoofing of private, or unallocated, or valid IP packets possible from these network locations

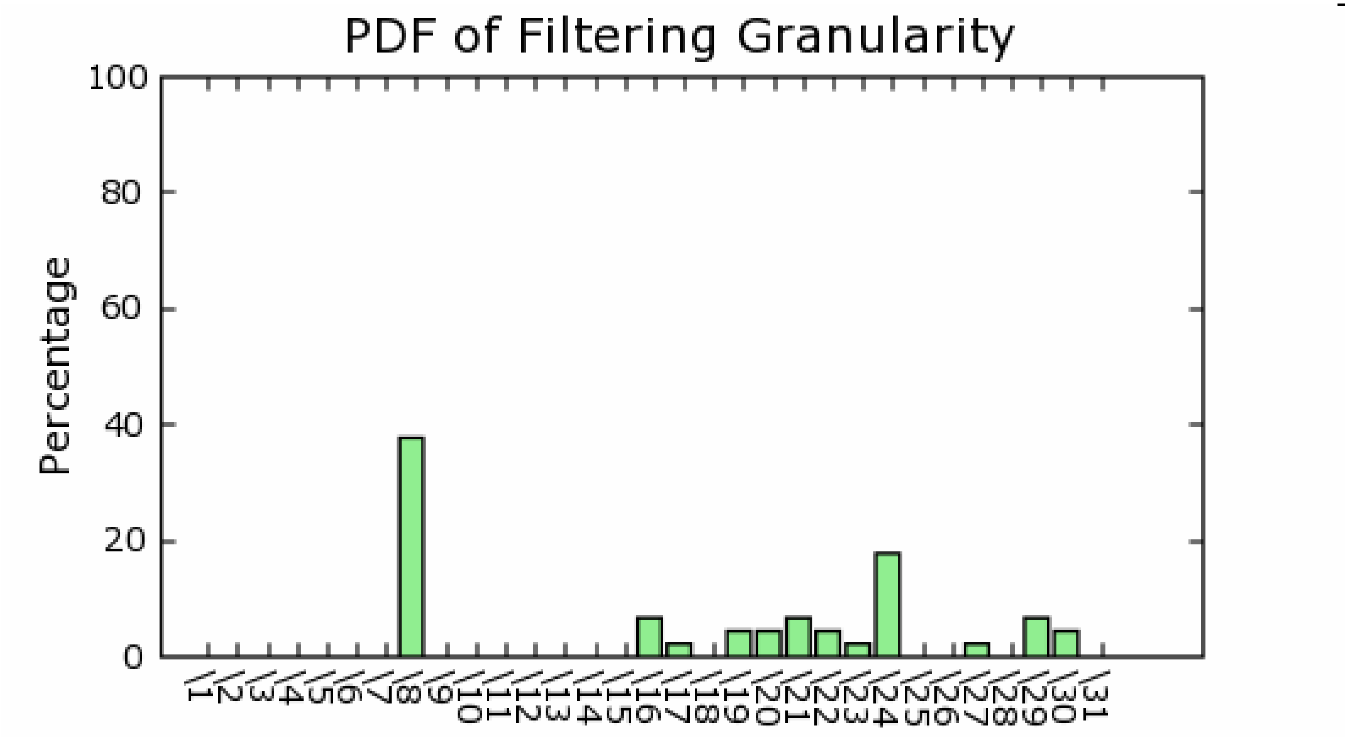
Filtering policies

Private	Unallocated	Valid	Client Count
✘	✘	✘	261
✘	✘		23
✘		✘	0
✘			59
	✘	✘	0
	✘		0
		✘	0
			0

✘ Filtered

🟨 Spoofable policies found in operation on the Internet

Filtering Boundaries

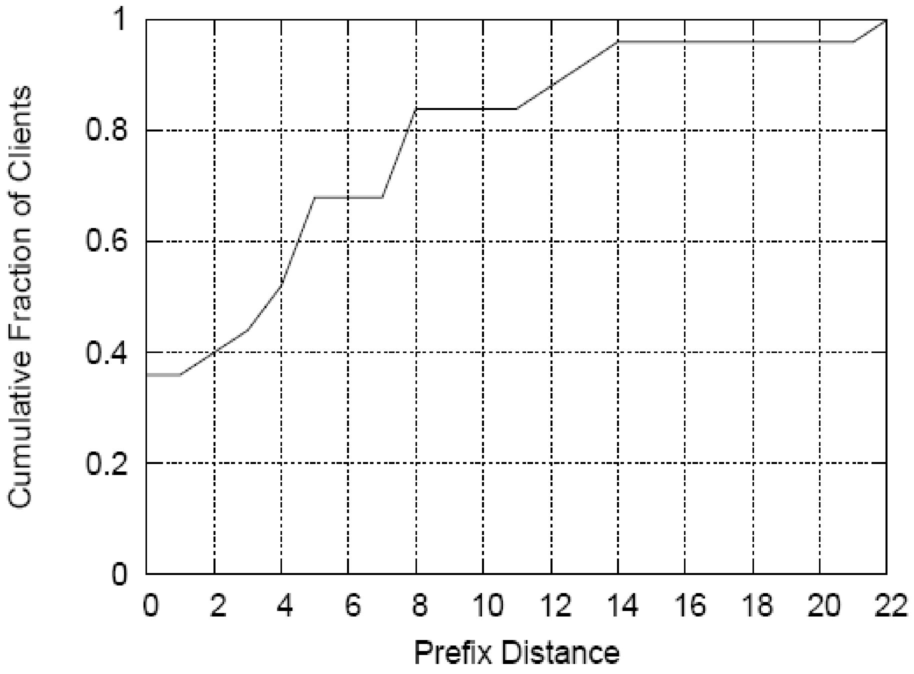


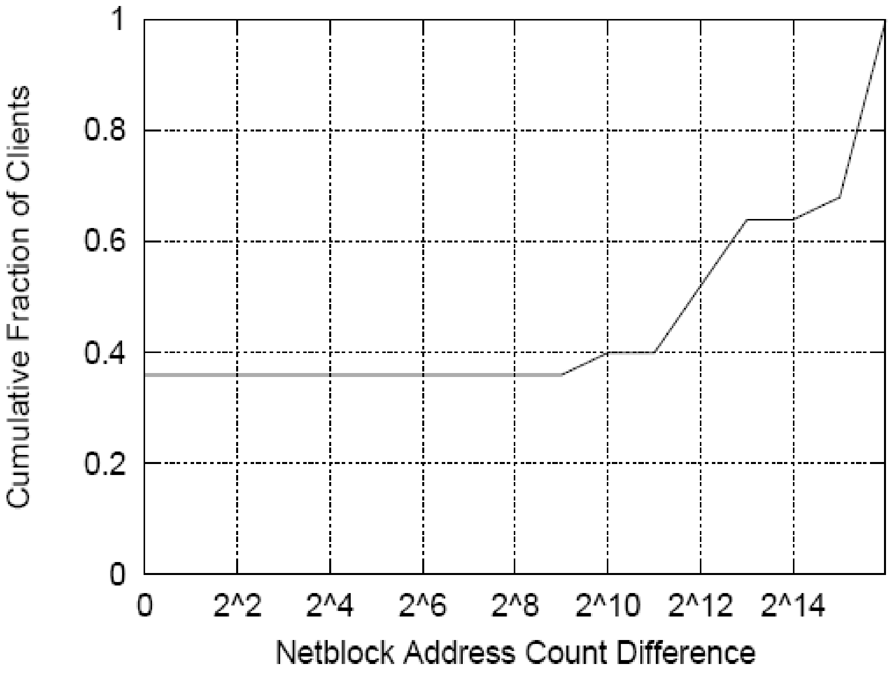
- Filtering occurring on a /8 boundary enables a client within that network to spoof 16,777,215 other addresses.

Correspondence between filtering granularity and BGP prefix size

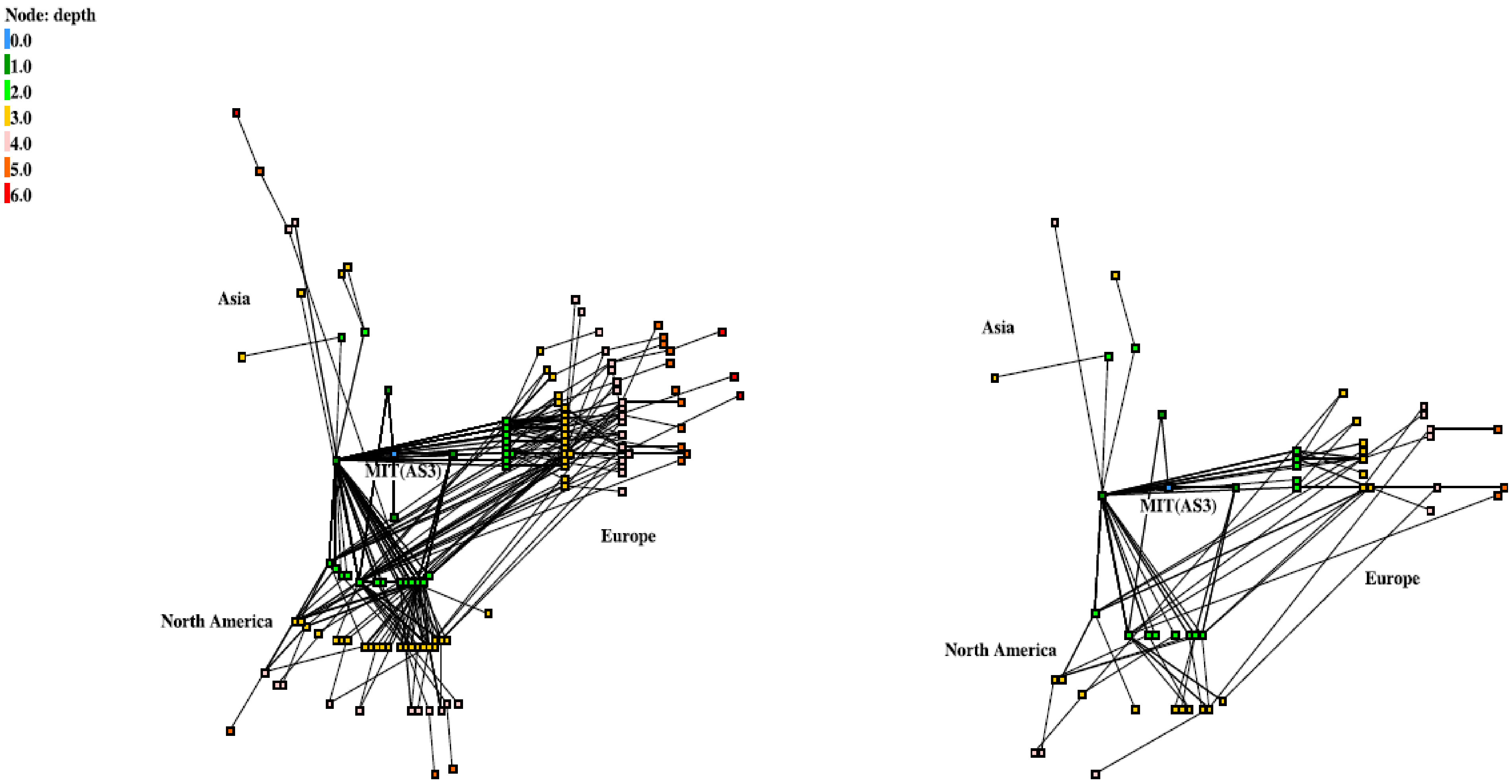
- **Important to understand how filtering granularity relates to routing announcements**
  - Are our extrapolations valid?
  - Provides clues to a provider’s network structure and operational practices.
- BGP view from University of Oregon Routeviews tables
  - prefix size
  - AS numbers

Correspondence between filtering granularity and BGP prefix size





- Over 36% of the time filtering boundary is exactly the same as announced netblock size
- Over 95% of the time within 65,536 IP addresses



(a) AS graph of all attempted spoof paths

(b) AS graph of spoofable paths

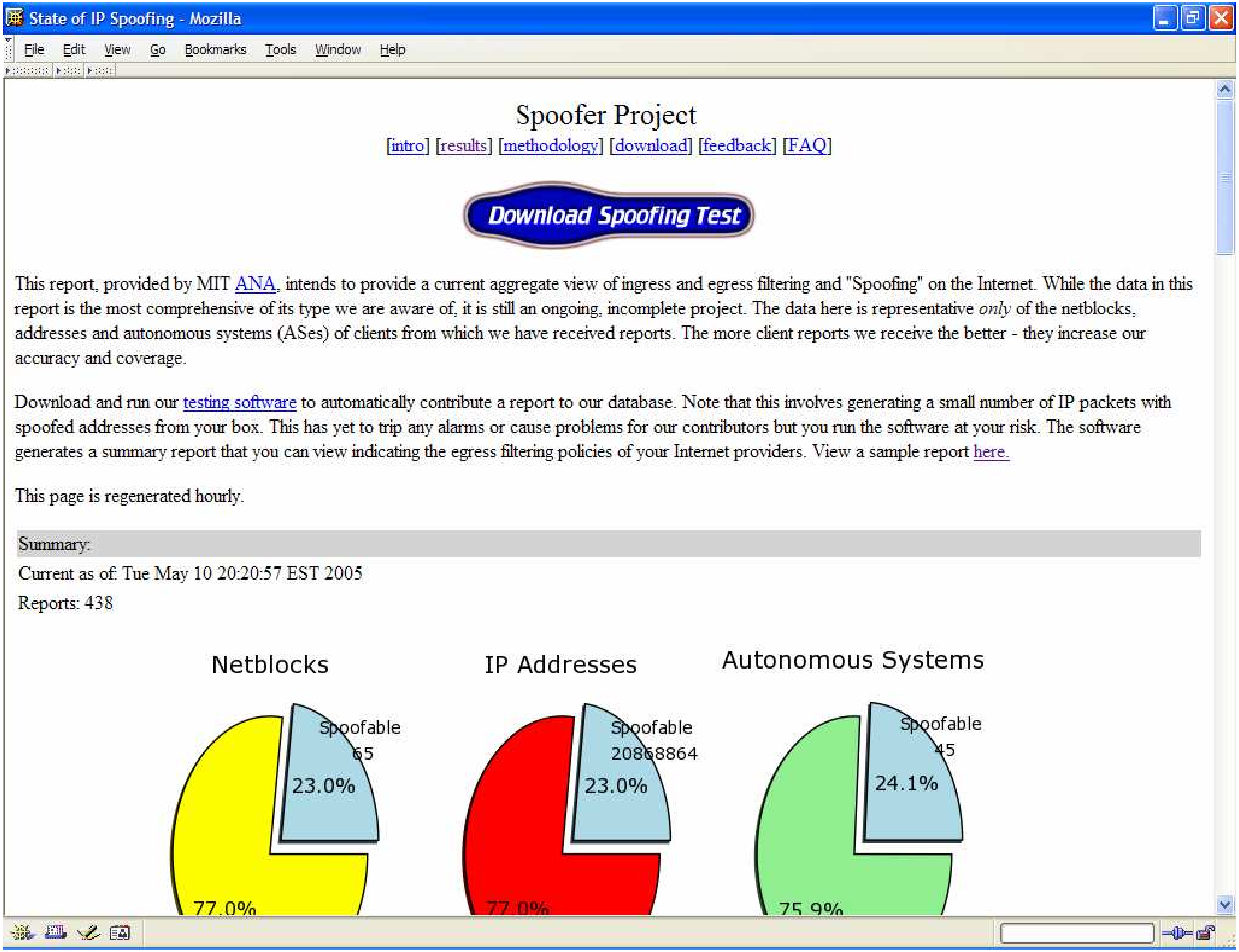
- **Spoofed packets that make it past the ingress edges are likely to travel across the entire Internet**
- **No geographic pattern to filtering policies**

Conclusion

Ongoing collection effort

http://spoofer.csail.mit.edu/summary.html

- Hourly-updated web page
- Summarizes current state of IP spoofing
- Goal: continue collecting reports to improve accuracy, detect trends, etc.
- We need help to expand coverage and gain more data!



Spoofer Project

[intro] [results] [methodology] [download] [feedback] [FAQ]

This report, provided by MIT ANA, intends to provide a current aggregate view of ingress and egress filtering and "Spoofing" on the Internet. While the data in this report is the most comprehensive of its type we are aware of, it is still an ongoing, incomplete project. The data here is representative *only* of the netblocks, addresses and autonomous systems (ASes) of clients from which we have received reports. The more client reports we receive the better - they increase our accuracy and coverage.

Download and run our testing software to automatically contribute a report to our database. Note that this involves generating a small number of IP packets with spoofed addresses from your box. This has yet to trip any alarms or cause problems for our contributors but you run the software at your risk. The software generates a summary report that you can view indicating the egress filtering policies of your Internet providers. View a sample report here.

This page is regenerated hourly.

Summary:

Current as of: Tue May 10 20:20:57 EST 2005

Reports: 438

http://spoofer.csail.mit.edu

Summary of key results:

- ~23% of observed netblocks corresponding to ~24% of observed ASes allow some from of spoofing
- Filtering policies corresponds reasonably well to netblocks announced in BGP
- Filtering is frequently applied inconsistently allowing spoofing of parts of the address space
- No discernable geographic pattern in address filtering policies

Thanks

Understanding the geographic distribution of filtering policies

- **Want to visualize:**
  - **Geographic distribution of paths**
  - **Extent of spoofing**
  - **Spoofable paths vs. all observed paths**
- Nodes: Map each client to its AS
- Edges: defined by AS path
- Semi-geographic coordinate system:
  - Similar to Skitter AS topology graphs
  - Our server at graph center (root)
  - Node radius: AS hop distance
  - Node degree: longitude of AS organization
- Using CAIDA's otter tool [Huffaker99] to build AS graph