The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet

Robert Beverly
MIT CSAIL
rbeverly@mit.edu

Steven Bauer
MIT CSAIL
bauer@mit.edu

Abstract

Forging, or "spoofing," the source addresses of IP packets provides malicious parties with anonymity and novel attack vectors. Spoofing-based attacks complicate network operator's defense techniques; tracing spoofing remains a difficult and largely manual process. More sophisticated next generation distributed denial of service (DDoS) attacks may test filtering policies and adaptively attempt to forge source addresses. To understand the current state of network filtering, this paper presents an Internet-wide active measurement spoofing project. Clients in our study attempt to send carefully crafted UDP packets designed to infer filtering policies. When filtering of valid packets is in place we determine the filtering granularity by performing adjacent netblock scanning. Our results are the first to quantify the extent and nature of filtering and the ability to spoof on the Internet. We find that approximately one-quarter of the observed addresses, netblocks and autonomous systems (AS) permit full or partial spoofing. Projecting this number to the entire Internet, an approximation we show is reasonable, yields over 360 million addresses and 4,600 ASes from which spoofing is possible. Our findings suggest that a large portion of the Internet is vulnerable to spoofing and concerted attacks employing spoofing remain a serious concern.

1 Introduction

The Internet architecture provides no explicit mechanism to prevent packets with forged headers from traversing the network. Malicious parties can leverage the ability to forge or "spoof" the source address of IP packets to mount various attacks. For example, the vulnerability of BGP routers to predictive, spoofed TCP resets was a major concern in the last year [5]. By spoofing the source address, an attacker or compromised host can send packets toward a victim anonymously. This anonymity greatly complicates the job of network operators trying to defend their networks. More insidiously, attackers with the ability to spoof can leverage reflectors [15], making distributed denial-of-service (DDoS) attacks particularly problematic.

Previous research investigates various means of mitigating spoofing. Jin et. al give a scheme to block spoofed packets based on hop count [11]. Bellovin proposes a probabilistic marking scheme to trace spoofed packets to their origin [2], while Snoeren suggests an efficient hash-based traceback mechanism [17]. Despite these research efforts, finding the source of spoofed packets remains an operationally difficult problem for network operators [7].

Techniques such as ingress and egress address filtering [6] and unicast reverse path forwarding checks [1] are employed in some production networks to prevent spoofing. However, current filtering practices are limited by multi-homing, route asymmetry, filter list maintenance and router design. Thus, filtering is generally practical only at the edge of the network and is not universally applied.

Is spoofing still a relevant issue? The rise of zombie farms, where spoofing provides little additional anonymity for an attacker, suggests spoofing may not be a useful technique. The proliferation of Network Address Translation (NAT) devices renders spoofing attacks from hosts behind the NAT useless as the IP header is rewritten. Despite these two factors, analysis of backscatter [13, 14] shows spoofing is still widespread.

Moreover, the difficulty in defending against spoofed attacks suggests that next generation attack farms may intelligently probe the network and adaptively change behavior based on the ability to spoof. Consider a 10,000 node zombie DDoS attack [3]. Assuming a worst case scenario where zombies are widely distributed, a network operator must defend against attack packets from 5% of the routeable netblocks. However, if 2500 of those zombies are attached to networks permitting spoofing, a significant volume of the traffic would appear to come

from any of the routeable netblocks. Not only would the attack be difficult to track and filter, spoofing complicates mitigation of the remaining non-spoofed traffic.

This paper presents an Internet-wide active measurement spoofing project. Clients distributed around the world attempt to test various filtering policies by sending a series of spoofed UDP packets. In contrast to backscatter analysis which observes only the result of spoofing and is oblivious to the identity of the true sources, our research is concerned with finding the portions of the network that allow spoofing. Our results are the first to quantify the extent and nature of source address filtering and the ability to spoof on the Internet. Our key findings are:

1. Approximately 24% of the observed netblocks, corresponding to 25% of the observed autonomous systems, allow spoofing.

2. Filtering is frequently applied inconsistently allowing for partial spoofing of portions of the IP address space.

3. No geographic region in our sample appears to be a predominate potential source of spoofed packets.

4. In over 36% of our cases, filtering policies are applied exactly on the routing boundaries observed in a global BGP routing table.

2 Methodology

All major operating systems require administrative privileges to send arbitrarily crafted packets as there are few legitimate reasons for spoofing. There is no way to coerce remote Internet hosts, to which we have no access, to send spoofed packets. Our testing therefore requires the cooperation of willing participants. We make publicly available a "spoofer" program, in source and binary formats. This program allows users to test their own network and records the results on our server. The software as well as continually updated summary statistics are provided on the Spoofer Project home page: `http://spoofer.csail.mit.edu`.

While our coverage is limited by the number of hosts which run the spoofer, we receive test reports from a non-trivial portion of the Internet. Section 3 examines coverage in detail.

2.1 Spoofer Operation

As depicted in Figure 1, in a spoofing test the client attempts to send a series of spoofed UDP packets to a server on our campus (step 1). As our server receives the packets, they are recorded in a database for later retrieval

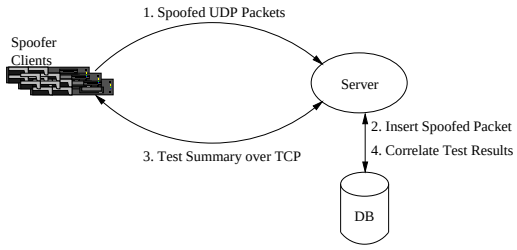


Figure 1: Spoofer test operation. Clients source a series of spoofed UDP packets to our server where they are subsequently disambiguated and analyzed.

Table 1: Source addresses of spoofed packets

Spoofed Source	Description
1.2.3.4	Unallocated
172.16.1.100	Private (RFC1918)
6.1.2.3	Valid (In BGP table)
$IP \oplus (2^N)$ for $0 \leq N \leq 24$	Neighbor Spoof

(step 2). Because UDP does not guarantee reliable delivery, the tester sends five packets for each source address with a random inter-packet delay between $(0, 500]$ms to prevent incorrect inferences due to loss.

After sending the spoofed UDP packets, the program establishes a TCP connection with the server to exchange test results and complete the client's test run (step 3). To avoid any secondary filtering effects that might drop test or report packets for reasons other than source address filtering, we use a UDP destination port of 53 for the spoofed packets and TCP port 80 for the test summary exchange. These ports correspond to the well-known DNS and HTTP ports respectively and should be open in the majority of circumstances[1].

The source addresses, summarized in Table 1, are specially chosen to test and infer common filtering policies. The first source address tested is as yet unallocated by the Internet Assigned Number Authority (IANA) [10, 8]. This address should not appear in any routing tables since IANA has not delegated it to any organization. Some networks employ filters that block traffic originating from these unallocated regions of the IP address space.

The second source IP address the program tests is one in a private netblock designated by RFC1918 [16]. Private IP addresses are often legitimately assigned to hosts in a private network for site-local routing, but should never appear on the public Internet. Best common practices dictate that privately addressed packets should be contained within AS boundaries.

Next, the spoofer sends packets with a valid, allocated

source address but one that is spoofed. In contrast to the first series of packets, this address appears in the global BGP routing table but is allocated to another organization. Notably, a filtering policy that allows only packets with source IP addresses that are present in the BGP table is easier for providers to implement and maintain.

Finally, the spoofer attempts to discover the granularity of any applied filtering by successively spoofing addresses from netblocks adjacent to its own. We accomplish this "neighbor spoof" by trying successively larger boundaries until spoofing an adjacent /8. To generate the address, we negate (i.e. flip) a bit at a time in the host's true source address beginning with the least significant bit. Thus, we start by spoofing an adjacent /31 address which corresponds to the host's address ± 1, i.e. the immediate neighbor's address.

For each source address, the spoofer includes a unique random 14 byte identifier string in the payload of the UDP datagram. The unique string allows us to later disambiguate which spoofed packets are received and which are blocked.

During the TCP connection, the client informs the server of its operating system and the number of spoofed sources it sent. For each source, the client reports the identifiers of the spoofed packets it attempted to send. Based on this transaction, the server records in the database the success or failure of the remote machine's ability to send spoofed packets (step 4). Finally, the spoofer runs a traceroute to our server which is also recorded in the database to determine the complete forward path of any spoofs.

At the end of the report transaction, the user is provided a unique URL pointing to a web page containing her test results. This web page summarizes the spoofing and filtering along the path from the user to our host.

2.2 Spoofer Run

For each client running the spoofer, several outcomes are possible. The spoofed UDP packets simply may not reach our server if a network filter or other policy blocks them. During the test summary transaction phase (step 3) of the spoofer, the server determines which packets the client believes it sent actually did not arrive.

In some cases the operating system may not allow sending spoofed packets even when running with administrative privileges. The most common instance of this restriction is Windows XP with Service Pack 2 (SP2)[2], however we see other operating systems with security restrictions that cause the client to fail. Alternatively, the packets may arrive with the true source address instead of the spoofed address if a NAT device rewrites the header.

Finally, some or all of the spoofed packets may arrive. In this case, the server will correlate the message identifiers the client reports with the identifiers of the UDP packets it received. In all cases, the server records the outcome in the database for later analysis.

3 Results

We advertised availability of the spoofer application on the North American Network Operators Group (NANOG) and dshield security mailing lists. Between March and May 2005, we received 570 client reports, 459 of which were unique [3]. All of our results exclude any data from machines belonging to our own campus netblocks and we count multiple reports from the same client IP addresses only once.

While the general user population may not be interested or motivated to run our spoofing test application, filtering policies are applied to netblocks and hence consistent over the size of that netblock. Because network routing advertisements are aggregated, the granularity of our netblock view may not exactly reflect filtering policies. Whether the netblocks coincide with globally advertised routing prefixes is a question we consider in Section 3.4. Therefore, for the purposes of coverage, it suffices if a single user is able to test a netblock. Unless there is direct evidence of the ability to spoof we classify netblocks as "believed unspoofable."

3.1 Failed Spoofs

In this subsection, we consider only the three types of spoofed addresses described in the methodology (unallocated, private and valid) and ignore neighbor spoofing for the time being. As previously discussed, all spoofing for a given client may fail for a variety of reasons. We do not infer anything about the netblocks of clients with these failures:

- *Socket creation blocked by Windows XP SP2:* 122 of the 216 reports from Windows (56%) indicate that the spoofer application was unable to create the raw socket and send spoofed packets. The only version of Windows which blocks packets with a spoofed source address is Windows XP running Service Pack 2 (SP2). That over half of the Windows machines are blocked implies that SP2 is widely adopted in our client population. Because of the dominant popularity of Windows as a client operating system [4], continued uptake of SP2 will have a significant positive impact on preventing spoofing from compromised hosts.

- *Socket creation blocked by other operating systems:* An additional 20 clients not running Windows were unable to send spoofed packets even when running

Table 2: Observed spoofing coverage

Metric	Spoofable	Believed Unspoofable
Netblocks	73	229
IP Addresses	21.0M	70.0M
ASes	52	150

Table 3: Spoofing coverage relative to observed and routeable space

Metric	Spoofable (% Observed)	Spoofable (% Routeable Space)
Netblocks	24.2%	0.04%
IP Addresses	23.0%	1.31%
ASes	25.7%	0.29%

with root privileges. These machines likely employ additional security mechanisms such as capabilities or had local packet firewalls that block the raw socket creation or sendto system calls.

- *Hosts behind NAT devices:* When the UDP packets arrive at our server but with a source address other than the source address they are sent with, the server marks them as rewritten by a NAT. We count all instances of NAT rewriting as a failed spoof since we cannot infer whether or not the host would have been successful had the NAT not been in place. 110 of the clients are behind a NAT device.

- *Totals:* 284 clients, or approximately two-thirds, failed to spoof any packets.

3.2 Spoofing Coverage

We quantify the coverage of spoofing along several dimensions. Using RouteViews [12] data, we determine the netblock and AS of each spoofer client. Based on the size of the netblock, we can determine the number of IP addresses the report approximately represents. Table 2 gives the number of netblocks, addresses and ASes that are spoofable and believed unspoofable as observed in our data set.

At the time of writing, there are approximately 1.59B globally routeable IP address, 18,000 autonomous systems and 169,000 netblocks. Table 3 presents the spoofing coverage relative to the addresses, ASes and netblocks we observed and the total global counts.

Approximately 24% of the observed netblocks, corresponding to 25% of the observed autonomous systems, allow some form of spoofing. Assuming a uniform distribution of testing, projecting these numbers to the entire Internet yields over 360M spoofable addresses and over 4,600 spoofable ASes. Our findings suggest that a large portion of the Internet is still vulnerable to spoofing, implying that concerted spoofing attacks remain a serious concern.

Table 4: Frequency of inconsistent filtering. Check marks indicate the presence of a particular filter.

Private	Unallocated	Valid	Instances
✓	✓	✓	229
✓	✓		21
✓		✓	0
✓			52
	✓	✓	0
	✓		0
		✓	0

3.3 Inconsistent Filtering

Many hosts experience inconsistent filtering where a subset of the spoofed packets arrive at our measurement station. We classify the different subsets of spoofing into these inconsistent filtering categories. Again, for the moment we ignore adjacent netblock spoofing. Table 4 summarizes the inconsistent filtering behavior we observe. A check mark in the table indicates the presence of a particular type of filtering, whereas no check mark indicates the absence of filtering.

- *Failed on private address (RFC1918), other spoofs successful:* 52 clients are able to spoof both valid and unallocated addresses, but are unable to spoof private addresses. Since blocking private addresses is an easy and time-invariant policy, it is unsurprising to find instances of only RFC1918 (“martian”) blocking.

- *Failed on private and unallocated addresses, valid (routeable) spoof successful:* 21 clients could not spoof the RFC1918 or the unallocated (“bogon”) addresses, but could spoof the valid address. This interesting class of inconsistency likely arises from policy that checks for a valid routing entry before forwarding packets. An advantage to providers in adopting this policy is that there is no need to continually and manually update packet filters as new addresses are allocated. An example of a community-wide project to provide a BGP feed of bogon routes to automate filtering is the Cymru bogon route-server project [18].

- *Failed on unallocated addresses, other spoofs successful:* We saw no inconsistencies of this type, implying that providers who are conscientious enough to block unallocated addresses also block RFC1918 packets.

3.4 Understanding Filtering Granularity

Next we turn to the problem of understanding filtering granularity. To determine the filtering granularity, we analyze the data from the adjacent netblock neighbor scan. Recall that the client attempts to spoof successive netblocks adjacent to its own, beginning with its immediate neighbor (i.e. a /31) and ending by testing an adjacent /8.

Figure 2 displays the granularity of either ingress or egress filtering employed by service providers tested in our study. If the filtering occurs on a /8 boundary for instance, a client within that network is able to spoof 16,777,215 other addresses. In our study nearly 40% of clients are able to spoof addresses upto a /8 netblock.

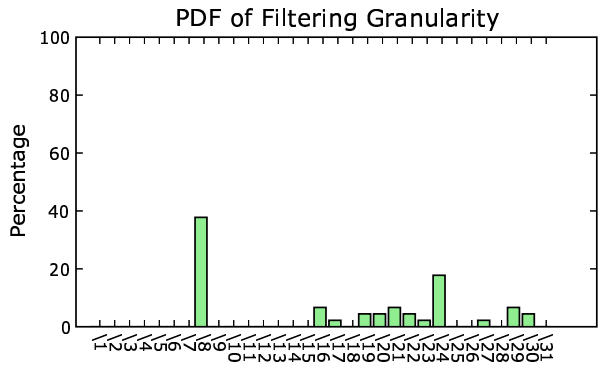


Figure 2: Probability distribution of filtering policy granularity as inferred by neighbor spoof scanning

We are also interested in how closely a host's inferred filtering boundaries match the size of its prefix in the global BGP routing table. This is important because our ability to infer the scope of spoofing is limited if filtering is applied at a very granular level, but routing announcements are highly aggregated. More generally understanding the correspondence between policy enforcement points and routing advertisements as seen from the global routing tables is useful knowledge in that inferences can be made about a provider's network structure and operational practices.

We define a "prefix distance" metric as the difference between the size of the advertised BGP prefix to which the client belongs and the maximum-sized adjacent neighbor netblock the client successfully spoofs. In cases where the client is able to spoof our other three test source addresses, the client is able to also spoof all possible neighbors. We ignore clients that are capable of full spoofing when analyzing filtering boundary effects.

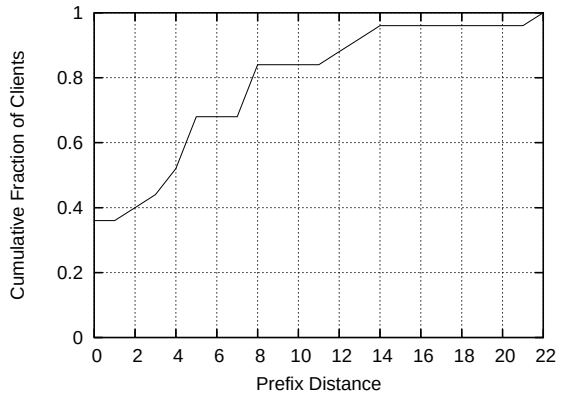
Figure 3(a) plots the cumulative distribution of clients as a function of prefix distance. Figure 3(b) examines the difference between the advertised and inferred address counts. Over 36% of the clients infer the same filtering prefix boundary as the actual route advertisement.

A final curious class of inconsistencies for which we have no good intuition is due to hosts that are able to spoof one of our sources, but are unable to spoof their immediate neighbor's address. In total we found two instances of non-neighbor spoofability. Although there are several possible explanations for this phenomenon, it is unexpected.

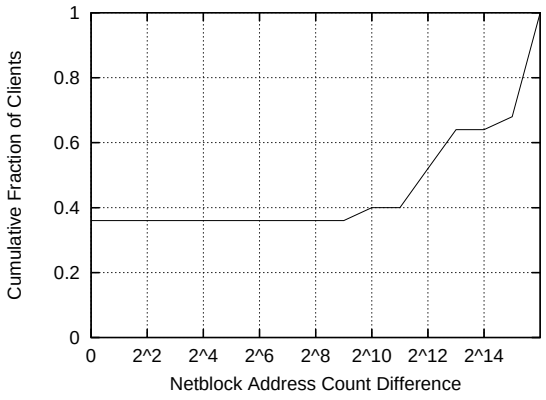
3.5 Visualizing the Extent of Spoofing

When our measurement host receives a spoofed packet from a client, we can definitively say that no device along the path from the client to our server performed any filtering to block the packet. To visualize the scope, boundary and geographic extent of spoofing, we use CAIDA's Otter tool [9]. The coordinate space places nodes with a radius corresponding to their AS path length depth from our measurement host and a degree representing geographic location of the AS. The images plot our measurement host in the center of the graph at depth zero labeled as "MIT(AS3)." For each client, we determine the AS path from the client to our measurement host and add each AS as a node in the graph. ASes are connected by edges according to the client's route toward the measurement host. The degree of each node is determined by the longitude of the organization responsible for the AS. In this fashion, we can visualize the geographic position of ASes as well as their BGP distance from the server.

Figure 4(a) contains all ASes of nodes from which we received test reports as well as the intermediate ASes on the path to our measurement host. We see several distinct clusters of nodes corresponding to Europe, Asia and North America. Figure 4(b) plots the ASes and paths corresponding to clients that successfully sent spoofed packets. The ASes are in the same coordinate system so that we can compare the two graphs and determine the boundary and range of spoofing as observed from our host. Comparing the two plots, no geographic area emerges with significantly differing relative ability to spoof. While the second graph is more sparse, a significant portion of the graph is spoofable. In general if filtering is not applied by the providers at the "edges," spoofed packets travel unabated across the Internet. Any filtering that is in place on the three major peering sessions our campus maintains had no effect in preventing spoofed packets from reaching our server.

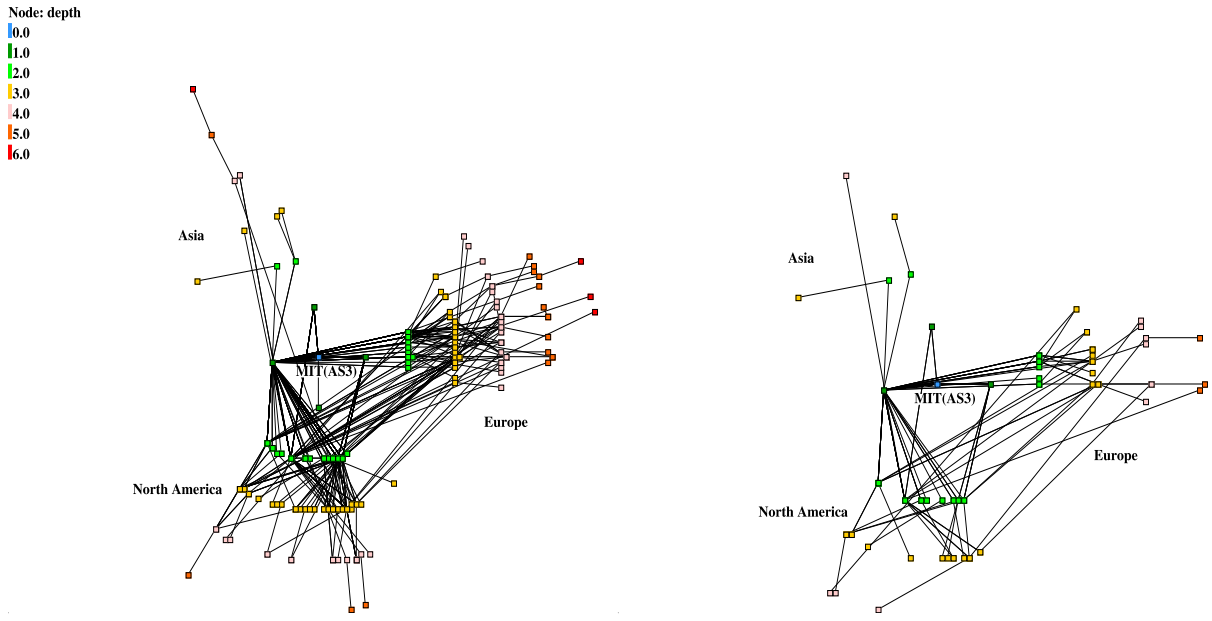


(a) Cumulative distribution of difference between advertised and inferred prefi x size.



(b) Cumulative distribution of difference between advertised and inferred netblock address counts.

Figure 3: Correspondence between filtering granularity as inferred by neighbor spoof scanning and BGP advertised prefixes in a global routing table.



(a) AS graph of all attempted spoof paths

(b) AS graph of spoofable paths

Figure 4: Visualizing geographic reach of spoofing. Each node represents an AS while edges show routing paths. The AS radius is determined by distance from our measurement host, degree by AS longitude. The graph of spoofable paths is a subset of all paths tested. Our server is in the center of the graph at depth zero.

4 Conclusions

To the best of our knowledge, this paper presents the first Internet-wide results examining the extent and nature of source address filtering. Our findings are germane to both network research and operational communities. We see that a significant fraction, approximately one-quarter, of the netblocks, IP addresses and ASes observed permit spoofing. This suggests that a large portion of the Internet is still vulnerable to spoofing and concerted spoofing attacks remain a serious concern. Projecting our results to the entire Internet yields over 360M spoofable addresses and 4,600 ASes from which spoofing is possible. This is particularly significant if next generation attack farms intelligently probe the network and adaptively change behavior based on the ability to spoof.

Currently, we are working on adding functionality to the spoofer that detects what point filtering is applied. In addition, we plan to test spoofing in the reverse direction; the ability to send spoofed packets to random Internet hosts. Combined with continued collection of spoofing reports[4], we hope this project serves as a significant step forward in understanding Internet filtering.

Acknowledgments

The authors would like to thank Mike Afergan, John Curran, Simson Garfinkel, Aaron Hughes, Ken Shores, Karen Sollins, John Wroclawski and countless NANOG members for constructive discussions and feedback. We also thank the maintainers of the RouteViews project for continuing to provide a valuable community resource.

References

[1] BAKER, F., AND SAVOLA, P. Ingress Filtering for Multihomed Networks. RFC 3704, Mar. 2004.

[2] BELLOVIN, S. M. ICMP traceback messages. IETF Internet Draft, Sept. 2000. `http://www.cs.columbia.edu/~smb/papers/draft-bellovin-itrace-00.txt`.

[3] BERINAT, S. Online extortion: How a bookmaker and a whiz kid took on an extortionist and won. *CSO Magazine* (May 2005).

[4] BEVERLY, R. A Robust Classifier for Passive TCP/IP Fingerprinting. In *Proceedings of the 5th Passive and Active Measurement (PAM) Workshop* (2004), pp. 158–167.

[5] DALAL, M. Improving TCP's robustness to blind in-window attacks. IETF Internet Draft, May 2005. `http://www.ietf.org/internet-drafts/draft-ietf-tcpm-tcpsecure-03.txt`.

[6] FERGUSON, P., AND SENIE, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000.

[7] GREENE, B. R., MORROW, C., AND GEMBERLING, B. W. ISP security: Real world techniques. NANOG 23, Oct. 2001. `http://www.nanog.org/mtg-0110/greene.html`.

[8] HUBBARD, K., KOSTERS, M., CONRAD, D., KARRENBERG, D., AND POSTEL, J. Internet Registry IP Allocation Guidelines. RFC 2050, Nov. 1996.

[9] HUFFAKER, B., NEMETH, E., AND K. CLAFFY. Otter: A general-purpose network visualization tool. In *Proceedings of INET* (June 1999), pp. 22–25.

[10] IANA. Internet Assigned Number Authority IP address allocations. `http://www.iana.org`.

[11] JIN, C., WANG, H., AND SHIN, K. Hop-count filtering: An effective defense against spoofed DoS traffic. In *Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS)* (2003), pp. 30–41.

[12] MEYER, D. University of Oregon RouteViews. `http://www.routeviews.org`.

[13] MOORE, D., VOELKER, G. M., AND SAVAGE, S. Inferring internet Denial-of-Service activity. In *USENIX Security Symposium* (2001), pp. 9–22.

[14] PANG, R., YEGNESWARAN, V., BARFORD, P., AND PAXSON, V. Characteristics of Internet Background Radiation. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference* (Oct. 2004).

[15] PAXSON, V. An analysis of using reflectors for distributed denial-of-service attacks. *ACM Computer Communications Review (CCR) 31*, 3 (2001).

[16] REKHTER, Y., MOSKOWITZ, B., KARRENBERG, D., DE GROOT, G. J., AND LEAR, E. Address Allocation for Private Internets. RFC 1918, Feb. 1996.

[17] SNOEREN, A. C., PARTRIDGE, C., SANCHEQ, L. A., JONES, C. E., TCHAKOUNTIO, F., KENT, S. T., AND STRAYER, W. T. Hash-based IP traceback. In *Proceedings of ACM SIGCOMM* (2001).

[18] THOMAS, R. Team cymru bogon route-server project. `http://www.cymru.com/`.

Notes

[1] If these well-known, well-used ports fail or are proxied, attempts to spoof almost assuredly fail as well under most reasonable network configurations.

[2] `http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/sp2netwk.mspx`

[3] Notably no users or network administrators reported any abuse as a result of running our spoofer.

[4] We continue to receive $\sim$10 reports per day and plan to exploit distributed testbeds for additional coverage.