Understanding the Efficacy of Deployed Internet Source Address Validation Filtering

Robert Beverly
MIT CSAIL
rbeverly@csail.mit.edu

Arthur Berger
MIT CSAIL
awberger@csail.mit.edu

Young Hyun
CAIDA
youngh@caida.org

k claffy
CAIDA
kc@caida.org

ABSTRACT

IP source address forgery, or "spoofing," is a long-recognized consequence of the Internet's lack of packet-level authenticity. Despite historical precedent and filtering and tracing efforts, attackers continue to utilize spoofing for anonymity, indirection, and amplification. Using a distributed infrastructure and approximately 12,000 active measurement clients, we collect data on the prevalence and efficacy of current best-practice source address validation techniques. Of clients able to test their provider's source-address filtering rules, we find 31% able to successfully spoof an arbitrary, routable source address, while 77% of clients otherwise unable to spoof can forge an address within their own /24 subnetwork. We uncover significant differences in filtering depending upon network geographic region, type, and size. Our new *tracefilter* tool for filter location inference finds 80% of filters implemented a single IP hop from sources, with over 95% of blocked packets observably filtered within the source's autonomous system. Finally, we provide initial longitudinal results on the evolution of spoofing revealing no mitigation improvement over four years of measurement. Our analysis provides an empirical basis for evaluating incentive and coordination issues surrounding existing and future Internet packet authentication strategies.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; C.2.3 [**Computer Communication Networks**]: Network Operations

General Terms

Measurement, Experimentation, Security

Keywords

Source address validation, IP spoofing, filtering

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

The Internet architecture includes no explicit notion of packet-level authenticity. A long-recognized [30] consequence of this weakness is the ability to forge or "spoof" IP packet headers. While willing and able networks implement various ad-hoc authentication techniques, history demonstrates that malicious users probe for, and capitalize on, any ability to spoof. A common attack vector is to spoof source IP addresses, to enable anonymity, indirection, and amplification exploits (e.g. [4, 33, 44]).

As good Internet citizens, many operational networks implement source address validation best common practices. Ingress address filtering [18, 45] and unicast reverse path forwarding (uRPF) checks [5] are effective against source spoofing. In practice however, implementation of such techniques is often limited by multi-homing, route asymmetry, lengthy ad-hoc filter list maintenance, and router design. More importantly, current anti-spoofing filtering techniques are hindered by *incentive and coordination* problems. A provider can follow all best practices and still receive anonymous, malicious traffic from third-parties who do not properly filter. Protection from spoofed traffic using existing practice requires global coordination, a difficult, expensive, and unenforceable goal. As a result, previous research [29, 34] and recent attacks [33] demonstrate that source address spoofing has remained a viable attack vector. Moreover, despite two-decade old exploits [7], new source spoofing based attacks continue to emerge; we review three in §2.

This paper seeks to understand the real-world efficacy of Internet source address filtering best practices. We leverage a widely distributed measurement infrastructure [21] in conjunction with active client measurements to facilitate this understanding. We tailor our probing of the network to infer the extent of different types of filtering. In addition, we develop and use *tracefilter*, a novel tool for determining the in-network location of source address filtering. We significantly extend an initial study [10] with the following new contributions and findings:

1. Use of the Ark [21] global distributed measurement infrastructure as active probe reception points. Ark facilitates path-based analysis and tomography over disparate (e.g. commercial, academic, etc) routes.

2. Analysis of $\sim 12,000$ unique tests which reveal significant differences between the filtering encountered by clients based on geographic region, network type, and network size.

3. 31% of clients are able to spoof an arbitrary, routable source address, indicating that attackers are likely to find a subset of compromised hosts from which to spoof. 77% of clients otherwise unable to spoof can forge an address within their same /24 subnetwork.

4. Existing filtering deployment handles the simple case of private addresses, but has difficulty with routable addresses. Of clients able to spoof, only ∼20% could spoof a private address, while 68% and 98% could spoof an unallocated and routable address respectively.

5. *tracefilter*, a source address validation filter location tool. Tracefilter finds 80% of filters a single IP hop from sources while 95% of blocked packets are filtered within the source's autonomous system.

6. Longitudinal analysis revealing no improvement in the deployment or efficacy of Internet best practices anti-spoofing filtering.

Our analysis provides an empirical basis for understanding the incentive and coordination issues surrounding existing techniques and informs strategies for Internet packet authentication in the future.

2. UNDERSTANDING THE THREAT

IP spoofing is the enabling force behind recent Denial of Service (DoS) attacks observed in operational provider networks, with some attacks consuming all available bandwidth on even 40Gbps links [4]. However, spoofing is no longer limited to simple DoS attacks, but is used in a multitude of ways. Here we illuminate three different recent exploits reliant on the ability to spoof source addresses. The diversity of new exploits attests both to the continued threat of spoofing-based attacks as well as the ability to spoof on the Internet.

2.1 DNS Amplifier Bandwidth Attack

Figure 1 illustrates a bandwidth-based DoS attack that relies on spoofing for reflection [35], amplification and anonymity. We assume the attacker finds or places a large TXT record in the Domain Name Service (DNS) system. The attacker sends spoofed DNS queries with the victim's source address to many public DNS servers. Each request queries for the large TXT record. The public DNS servers retrieve the record and send it to the spoofed IP address, i.e. the victim. Thus, each small DNS query packet from the attacker is amplified and the victim cannot identify the true source. Because the result is cached by the intermediate DNS server, the attacker can continually query the servers and generate a DoS attack from innocent third parties.

This attack is difficult to defend against since the third-party DNS servers cannot distinguish legitimate requests from spoofed ones and traffic filtering is impractical against DNS traffic. Further, the attacker can find or dynamically create many different domains to evade detection. Such DNS attacks[1] have been seen in the wild by the operational community [32].

[1]Additionally, the DNS recently experienced widespread cache poisoning attacks reliant on IP spoofing [44].

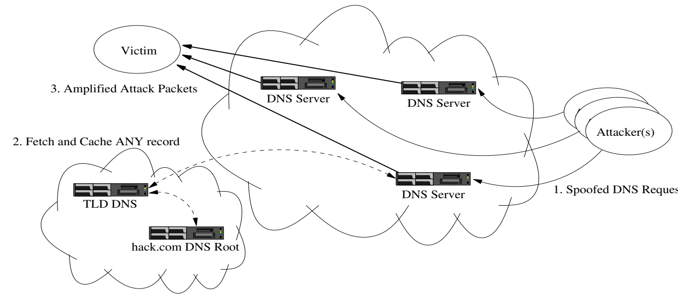


Figure 1: DNS Amplifier: 1) Attacker spoofs DNS request with victim's source for large TXT record. 2) Third-party DNS servers fetch and cache record. 3) Server responds to victim. Attacker's small query packets are amplified and victim cannot identify attack source.

2.2 In-Window TCP Reset Attack

A recent, non-bandwidth attack uses spoofed source TCP reset packets [43]. A TCP stack that receives a reset with a sequence number within its current window terminates the connection. Typically sessions are protected from third-party resets through port obfuscation, short duration and small window size. However, high-speed links with large bandwidth-delay products and well-known, long-lived flows creates a situation where an attacker can randomly find a valid in-window sequence number. Such attacks can disrupt persistent tunnels and even IP routing.

2.3 Spam Filter Circumvention Attack

Due to the widespread prevalence of worms, viruses and bot-farms used to send unsolicited commercial email, providers often prevent hosts on their network from establishing random SMTP (TCP port 25) connections and force user authentication with their own Mail Transfer Agent (MTA). The final new attack we mention does not disrupt the network but is instead a clever means to circumvent such provider-based spam filtering. Known as "fantasy mail," this attack has been repeatedly observed on production networks [31].

In Figure 2, we assume the spammer controls a zombie. Because the zombie's provider filters outbound SMTP SYN traffic, control of the zombie provides no additional advantage to the spammer. Instead, the spammer finds a second network that permits spoofing; recent attacks use dialup. The spammer sends a TCP SYN to port 25 of a public MTA with the zombie's source address. The zombie forwards the SYN-ACK from the MTA to the spammer over a direct TCP connection or other back-channel. The spammer can then send the correct spoofed packet to complete the TCP three-way handshake. By repeating for all packets, the spammer circumvents the provider's filter and spam appears to originate from the zombie. In this case the filters confuse network operators who cannot determine how the zombie sent the spam.

This example highlights the fact that seemingly straight forward filtering rules are exploitable – the operator assumed that the stateless, unidirectional filter sufficed to prevent a successful SMTP TCP connection. A second filter rule blocking incoming traffic with source port 25 defends against this specific asymmetric traffic spoofing attack. However,

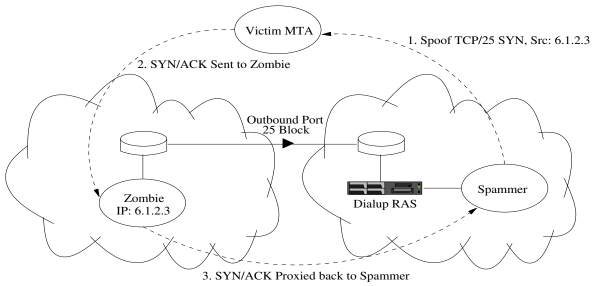


Figure 2: Circumventing Filters: spammer controls zombie on a network that filters outbound TCP port 25 SYNs. 1) Spammer spoofs SYN from dialup network with the zombie's source address. 2) MTA replies to zombie with a SYN/ACK. 3) Zombie forwards the MTA's packets back to the spammer.

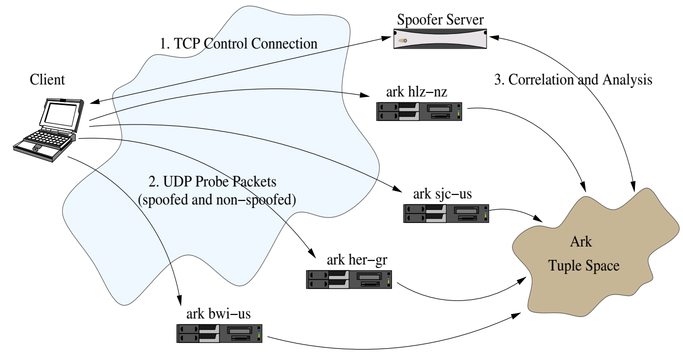


Figure 3: Spoofer test operation: 1) Clients receive test scenario from control server. 2) Scenarios involve sourcing a series of spoofed and non-spoofed UDP probe packets to ark nodes across the Internet. 3) Control server disambiguates and analyzes the results.

real-world deployments often involve distinct inbound and outbound SMTP servers, necessitating more rules. In order to cover all possible corner cases and exceptions, these rules quickly become complex, error-prone, and difficult to keep updated. The cost of a false filtering, where valid traffic is accidentally filtered, is quite high. Thus, providers often elect to simply forgo even simple source validation filtering.

In sum, network operators cannot reliably anticipate or defend against the next exploit or shift in attack patterns that leverage spoofing.

3. METHODOLOGY

Our methodology is based on a public "spoofer tester" application we built that allows voluntary users to test their Internet provider. The IP Spoofer Project page, `http://spoofer.csail.mit.edu`, hosts the software and provides continually updated statistics[2].

3.1 High-Level Operation

Our measurement infrastructure includes a single control server and approximately 30 globally distributed reception endpoints. Each endpoint is a member of CAIDA's Archipelago ("ark") project [21]. The distribution of ark nodes on various commercial, academic, and research networks around the world enables study of disparate paths with potentially different policies.

An invocation of the test begins with the client establishing a TCP connection with our control server as depicted in Figure 3 (step 1). After handshaking, the server instructs the client to perform a series of probes specified in a test scenario. A test scenario contains (*src_ip*, *dst_ip*, *timestamp*, *id*, *hmac*) tuples. The *src_ip* and *dst_ip* fields specify respectively the source and destination IP address to use for the probe. The source addresses are chosen to infer specific best-practice filter types and policies, described in §3.2. Destination addresses are drawn from the set of ark nodes with available testing capacity. The *id* is a unique random 14-byte probe identifier. Finally, the timestamp and Hash Message Authentication Code (HMAC) allow the ark nodes to authenticate the validity of an incoming probe and prevent replay attacks against the measurement infrastructure.

[2]Beyond its research component, many users employ the tester as a diagnostic tool for securing their own networks.

Test probes use UDP destination port 53 while the TCP control connection uses port 80. These ports correspond to the well-known DNS and HTTP ports respectively. While these ports are typically open, the tester detects any port blocking in order to ensure that no secondary filtering effects pollute the measurements.

As part of the test scenario, the client sends non-spoofed UDP packets with a fixed initial time-to-live (TTL). The purpose of the initial round of legitimate UDP packets is two-fold: first to guarantee that traffic can pass on UDP port 53 unimpeded and second to measure the path length. In the event that non-spoofed packets are blocked or sent through a transparent proxy, the test terminates.

Using the addresses specified in each test scenario tuple, the client next attempts to send a series of spoofed UDP packets to each ark receiver (step 2). The client copies the corresponding identifier, timestamp and HMAC into the probe's payload. To mitigate potential loss or bias, the test sends five packets for each source and destination address pair with a random inter-packet delay between $(0, 500]$ms. Ark nodes receive and validate incoming UDP probes, publishing valid test data into the ark tuple-space.

Each probe faces one of five possible outcomes. First, the operating system may prevent the application from sending the packet. While we largely mitigate operating system restrictions by falling back to raw Ethernet as necessary, a small number of configurations preclude even low-level spoofing. The tester detects instances of operating system blockage by trapping all failed calls to send traffic on the raw socket; this check detects local firewall rules, RPF checks, and protected socket options. Instances of operating system blocking are communicated to our server and recorded.

Second, a network address translation (NAT) device along the path may rewrite the packet header with a different source address or enforce a strict binding between link-layer and IP addresses. We explicitly detect rewritten packets and clients behind NATs. Third, the packets may be blocked by an in-network source address validation filter. Fourth, the packet may be dropped for reasons other than a spoofed address, such as congestion, and hence multiple copies are sent. Finally, the packet may successfully arrive at the intended recipient.

Table 1: **Source IP addresses designed to infer common operational filtering techniques**

Source IP	Description	Possible Defense
172.16.1.100	Private (i.e. RFC1918)	Static ACL
1.2.3.4	Unallocated	Bogon Filters
6.1.2.3	Valid (i.e. in BGP table)	uRPF
IP $\oplus(2^i)$ for $0 \leq i \leq 24$	Neighbor Spoof	MAC bindings, DOCSIS[14], etc.

From the perspective of combating source-address spoofing, or, equivalently, of an adversary attempting to send spoofed traffic, one is interested in any of the locations at which defenses can be placed, including the operating system, NATs, network firewalls, etc. From the more narrow perspective of the challenges faced by network operators, we are interested in spoofed packets that reach the operator's network. This paper focuses on the latter perspective. Thus, we exclude from the analysis clients behind a NAT and clients whose spoofed packets are blocked their operating system. Complete details of excluded clients are given in the Appendix.

However, as mentioned above, our methodology detects such clients, and to provide some perspective on their presence in the data set, in §5.4, Figures 10(a) and 10(b), we include them as separate categories.

The control server asynchronously retrieves and stores results from the tuple-space. Probe identifiers facilitate later disambiguation between received and blocked packets. After sourcing spoofed probes, the test client performs a tracefilter run, described in §3.3. The client sends a final round of non-spoofed packets to ensure path consistency throughout the duration of the test.

Testing ends with the client exchanging test results with the control server over the reliable TCP connection. The client reports its operating system, probe packet identifiers, and a traceroute of the forward path to each ark destination. The server correlates the packet identifiers to determine which are received or lost (step 3). The server infers the filtering or lack of filtering as the difference between the two sets.

3.2 Detecting Filter Types

We specifically design the test probes in such a way as to allow inference on the presence of various best practice filtering methods as shown in Table 1.

A trivial source to defend against is a private [36] addresses. Private IPs allow for site-local addressing, but should not be routable on the Internet. Therefore best practice policy dictates a static access control list (ACL) filter to block packets with private RFC1918 source addresses on ingress and egress.

The next source is as yet unallocated by IANA [22]. This address should not appear in routing tables since it is not delegated[3]. Network providers may maintain static ACL lists (so called "martian" filters) to block such bogon addresses, however this kind of manual maintenance is cumbersome, expensive, and error-prone given the number of network devices and churn in delegated address space. A popular automated alternative to static bogon filters is a bogon BGP feed [42] allowing providers to blackhole traffic for the feed's advertised destinations.

[3]While unallocated addresses appear in various routing table views, we verify 1.2.3.4 is not routed against routeviews.

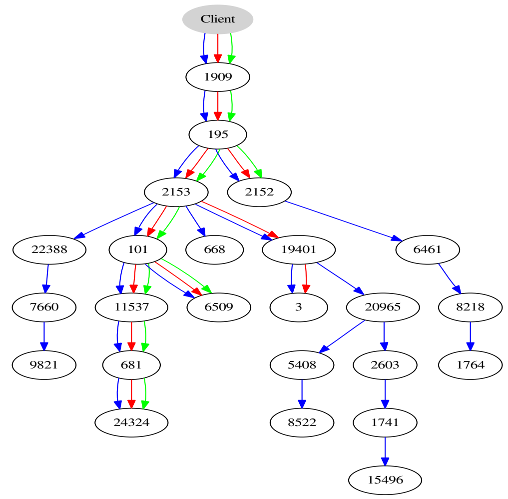


Figure 4: Example test result: edge colors correspond to types of spoofed traffic passing between ASes. Users can visualize locations of various types of filtering.

Spoofed valid addresses, i.e. those allocated and appearing in the global routing table, are more difficult to detect. Best practice [5] strategies dictate the use of either strict or loose uRPF. Strict uRPF finds the outgoing interface to reach the source of an incoming packet. If the outgoing interface is different from the interface on which the packet arrives, the packet is dropped. Unfortunately, strict uRPF is typically impractical given route asymmetry [17]. Instead, various modes of loose uRPF permit only packets with source IP addresses present in the routing table. Such filters are simple to implement and do not require periodic maintenance, but may be unable to block all spoofed traffic.

Finally, the tester attempts to discover the granularity of any filtering by successively spoofing addresses in adjacent netblocks. This "neighbor spoof" tries successively larger boundaries until spoofing an address in an adjacent /8. To generate the address, we exclusive-OR the host's true source address with 2^i for $0 \leq i \leq 24$ where i is size of the subnetwork. Intuitively we are simply negating (i.e. flipping) one bit at a time 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.

The combination of multiple ark destinations and our filter type inference enables us to provide users with a graphical picture of their network filtering. For example, Figure 4 depicts the autonomous system (AS) paths probed in the course of one test; the digraph's leaves correspond to Ark destinations. Colored directed edges indicate different types of spoofed source traffic passing between ASes. In this example, ASes 22388, 668 and 6461 block private sources, whereas other ASes block bogon sources. Beyond intuitive value to users, the Ark destinations permit path-based tomography.

3.3 Tracefilter

Conventional wisdom dictates that ingress filtering is performed near the edges of the network rather than the core. In addition to the nature and extent of IP spoofing, we also locate where in the network filtering is employed with a new, novel technique we dub *tracefilter*. In the same spirit as traceroute [23], tracefilter depends on TTL expiration and ICMP expiration notices.

Each router along a forwarding path decrements the time-to-live (TTL) value in the IP packet header. When the TTL reaches zero, or "expires," the router stops forwarding the packet in order to prevent routing loops. Routers generate an ICMP TTL exceeded message [12] back to the source of the packet once it expires[4].

Tracefilter works in conjunction with a control server we maintain, whose address we denote S. An invocation of tracefilter on a client C sends spoofed UDP packets with a source address of S and TTLs from $1, \ldots, d$. In this way, our server receives and processes any ICMP messages triggered by the packet[5]. Tracefilter probes use destination address $S+1$, an IP address on the same subnetwork as the control server. While the destination address need only be a valid IP address, we use $S+1$ to test a valid, congruent path.

As spoofed source packets are sent into the network, those that are not blocked by a filter elicit ICMP TTL exceeded messages sent to S. Note: routers, at least Cisco routers, first execute the filter and then test the TTL – the design principle is that input filters are applied prior to forwarding, and TTL processing is part of the forwarding process, [2]. An example of a tracefilter run is shown in Figure 5 and where a source address validation filter is present at the third hop. When Tracefilter sends packets with $ttl = 1$ and $ttl = 2$ along the path from C to S, and the first hop router and the second hop router, respectively, generated ICMP TTL exceeded messages to S. S can infer that the spoofed packet was not filtered at the first two hops. Tracefilter then tests the third hop for filtering, and so sends spoofed source packets with $ttl = 3$. A source address validation filter at the third router hop determines that S belongs to a different portion of the network, therefore any packets with source S should not have originated from the network to which C is attached and hence should be dropped. No further processing of the packet is done, and thus no ICMP TTL exceeded message is sent. Hence, in this example, 2 is the highest TTL of packets that elicit an ICMP TTL exceeded message, and at the third hop is the router that filters the spoofed packet. In general, the highest TTL of packets that elicit an ICMP TTL exceeded message is one less than router hop that filters the spoofed packets. If the spoofed packets are filtered at the first router hop, then no ICMP TTL exceeded message is sent.

How can S reliably determine the originating TTL of packets C sends? ICMP TTL exceeded messages include only the first 28 bytes of the original packet (the IP header plus the first 64 bits of payload) [27]. Thus, the ICMP message quotation includes only the IP and UDP headers of the packet that triggered the message, neither of which include the original TTL of the triggering packet.

[4]Some routers disable the diagnostic functionality ICMP messages afford to prevent specific or anticipated attacks.

[5]While tracefilter is a measurement utility, a malicious party could spoof in this way as an ICMP DoS technique.

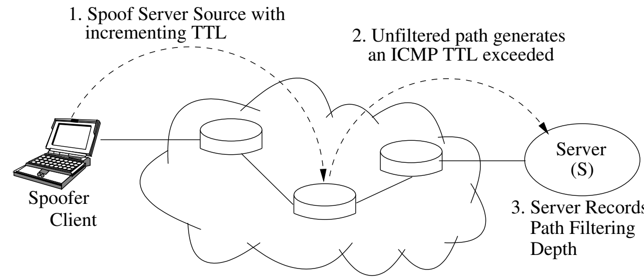


Figure 5: Example of *tracefilter* second iteration: 1) Client sends packet with TTL=2, src=S, dst=$S+1$, payload length=2. 2) With no filters along first two hops, the packet expires, generating an ICMP TTL exceeded message to our server S. 3) The server records the originating TTL of 2 by decoding the ICMP message body's UDP length. Client tests entire path; the largest originating TTL is one less filtering point.

IP			UDP		Payload
SRC: S	DST: S+1	TTL: 3	SRC: SessID	DST: 53	000

ICMP	IP			UDP	
Type: Time Exceeded	SRC: S	DST: S+1	TTL: 0	SRC: SessID	Len: 11

Figure 6: Tracefilter packet format and resulting ICMP TTL exceeded messages. To encode an originating TTL of 3, the tracefilter UDP packet contains a three-byte payload. To decode the originating TTL from the ICMP message, S extracts the UDP length from the ICMP quotation and computes $TTL = Len - 8$.

The TTL of the packet which triggers the ICMP TTL exceeded message is by definition zero. To encode the originating TTL, we pad the payload of the tracefilter packets so that the UDP length field (len) encodes the originating TTL value. For example, if the originating TTL is three, $len = 8 + 3 = 11$, as the UDP header is eight bytes. Our server then recovers the originating TTL of each tracefilter packet by decoding the UDP length field contained within the ICMP message body. By recording the largest received TTL from a client, the server can infer the number of hops along the path from the client to our server where spoofing filtering is employed. Figure 6 shows the format of the tracefilter packets and the ICMP TTL exceeded messages they generate.

A final detail is how C determines the maximum TTL, d, to test. The client could use a fixed maximum TTL or attempt to infer the distance itself via traceroute, but we require a more principled approach. Instead, tracefilter begins by measuring the IP path length between C and S. Tracefilter sends non-spoofed UDP packets with a TTL set to 64 so that our server can infer the IP hop length of the tested path. S extracts the TTL value from the received UDP packets and computes a distance $d = 64 - TTL_{recv}$. The server then communicates this distance d to the client over the TCP connection. At the conclusion of the test, a second round of non-spoofed UDP packets are sent to ensure that the path length has not changed due to multi-path routing or topology changes. We ignore instances of tracefilter where d shows inconsistency[6].

4. DATA SET

This section details our data set and an analysis of its generality as compared to the Internet population. The raw anonymized data collected in this study is publicly available from: `http://spoofer.csail.mit.edu/data/`.

4.1 Data Set Views

Over the 50 month period between February 2005 and April 2009, we received 19,474 client reports, 12,463 of which are unique clients[7].

The various analyses we perform require different subsets of this full population. For instance, we often wish to separate local filtering effects from network filtering, determine the effect of including a larger set of destinations, include only those clients which could (or could not) spoof a single type of address, understand temporal effects, etc. As such, there are multiple "views" of our data. For a formal description of the data view used in each experiment, the reader is directed to the Appendix.

The Internet's structure and policy naturally evolves over the course of data collection. Where possible, our analysis is inclusive of the entire 50 month period, and when not, we explicitly indicate sub-periods. In particular, our longitudinal results focus on two periods with the largest number of tests: a three-month window surrounding the 2005 coverage of the project in Slashdot [11] (approximately 1700 tests) and a comparable time window corresponding to our addition of the 30 CAIDA ark [21] destination nodes in February 2009 (approximately 800 tests).

4.2 Comparison of client set with general population

As the clients in our data set are self-selected and represent a relatively small number of distinct IP addresses, about 12,500, it is worthwhile to compare the set of clients with an estimate of the global population to provide a sense of how representative the former is of the latter.

For an estimate of the global population, we gathered IP addresses from a series of CAIDA topology traces. These topology traces were generated by distributed traceroutes, also using Ark, that attempt to trace the route to a destination in each routed /24 address prefix in a "team probing cycle." The union of cycles in May 2009 provides approximately 20.8M unique IP addresses of which 2.3M are likely intermediate routers. Because only intermediate traceroute hop replies are verifiably routers, this imposes an upper bound of 18.6M host addresses. We aggregated these 18.6M address to unique /24's, and compared attributes of these addresses with those in the measurement set on a per-address basis and a per /24 basis. The aggregation at the /24 granularity allows us to determine if the hosts within that prefix have homogeneous properties such as geography, speed, etc.

Table 2: Distribution of host addresses by geography, connection speed, and domain

Continent	General Population		Measurement Set	
	[per IP]	[per /24]	[per IP]	[per /24]
N. America	37%	34%	36%	37%
Europe	29%	29%	33%	34%
Asia	28%	30%	17%	17%
S. America	4%	4%	4%	4%
Oceania	1%	2%	2%	2%
Africa	0.5%	0.7%	6%	4%

Country	General Population	Measurement Set
	[per /24]	
United States	30%	32%
China	9%	1%
South Korea	7%	0.5%
Japan	6%	1%
Germany	6%	4%
U.K.	4%	5%
France	3%	2%
Italy	3%	4%
Canada	3%	4%
Spain	2%	1%

Connection Speed	General Population	Measurement set
	[per address]	
Broadband	68%	52%
xDSL	18%	24%
Cable	10%	16%
Other	4%	8%

Domain	General Population	Measurement set
	[per address]	
.edu	2%	4%
.org	1%	1%
other	76%	81%
unknown	21%	14%

To understand how representative our data set is as compared to the general population, we used NetAcuity [1] to determine geographic and line-speed attributes. Table 2 contains results of this comparison. The percent of addresses (/24's) in the general population and in the measurement set that are in North America is rather close, 37% and 36%, (34% and 37%). And thus, the percent of all clients outside of North America is close too; though the measurement set is somewhat over-weighted with clients in Europe and under-weighted for clients in Asia. Note: "client" refers to a host IP address, as opposed to an individual or a household.

The measurement set is quite diverse geographically, containing addresses from 132 countries. Examining the ten countries with the largest fraction of /24's, the measurement set is significantly light on addresses in China, South Korea, and Japan (it is over-weighted for addresses in India), though it is quite close for the listed European countries.

Regarding the connection speed, the category "broadband" refers to a non-determinant high speed connection, i.e. one believed to be high-speed, but whether it is, for example, cable, or Fast Ethernet or OC3 is not determined. With-

[6]One could guarantee path stability by examining the sources of ICMP messages on a hop basis. However, we are interested in aggregate statistics and our path-length check covers the vast majority of cases.

[7]We received no reports of alarms or abuse, illustrating both the difficulty of, and apathy toward, preventing spoofing.

out judging the accuracy of the NetAcuity determination of connection speed, at least in the present context, the measurement set is not qualitatively different from the general population.

A final concern with the self-selection of clients is whether a disproportionate number are attached at universities or research organizations (where possibly the access ISP's of these institutions is not representative). We classify addresses into four categories: .edu, .org, some other domain, or unknown (i.e. could not be determined). Although the proportion of clients in the measurement set determined to be on .edu, 4%, is higher than the general population, it is still small. The proportion on .org is also small and matches that of the general population. Regardless of the true distribution of addresses whose domain is unknown, we can say that at least 81% of the clients belong to a domain other than .edu or .org.

Overall, we are pleased with the diversity of the measurement set and how well its distribution matches the general population. While our data naturally suffers from sample bias, we believe our results are sufficiently general to accurately support the main points of the paper.

5. RESULTS

This section analyzes our results along four dimensions: a) prevalence and effectiveness of different types of best-practices filtering; b) granularity at which filtering policy is employed, and thus the ability to spoof neighboring addresses; c) location of filtering within the network; and d) variations in results dependent on origin network size, geographic location, class, etc. Lastly, we discuss tests which we remove from analysis due to secondary effects, e.g. NATs.

5.1 Effectiveness of Filtering

We first examine the efficacy of existing Internet filtering on the basis of our probes specifically crafted to infer the existence of different filtering techniques. Recall that clients send spoofed source probes with valid, bogon, and private sources (§3.2) as given in Table 1. Note, all analyses within this paper treat multiple reports from the same client IP address within any given period only once.

In the most recent three-months of data collection, 31% of all clients are able to send traffic from one of the three main source addresses to at least one of our test destination receivers. This result is significant in comparison to our earlier findings of between 20% and 30% of clients, netblocks, and ASes capable of sourcing spoofed traffic [10]. These most recent findings reinforce the contention that little additional progress has been made in deploying filtering.

Which sources are most effectively spoofed through the Internet, or conversely, which are most frequently blocked? For the most recent three-month period, where the CAIDA Ark receivers serve as possible probe destinations, Figure 7 depicts the cumulative fraction of clients capable of spoofing as a function of the fraction of Ark destinations receiving each type of spoofed source probe. For example, 90% of clients could spoof a bogon address to 40% or fewer of the destinations (including zero destinations).

Note that an increase in the set of possible destinations could only increase the percent of clients that could spoof toward at least some destination. Thus from this perspective, the values reported in Figure 7 are conservative.

The y-intercepts on Figure 7 indicates the fraction of clients

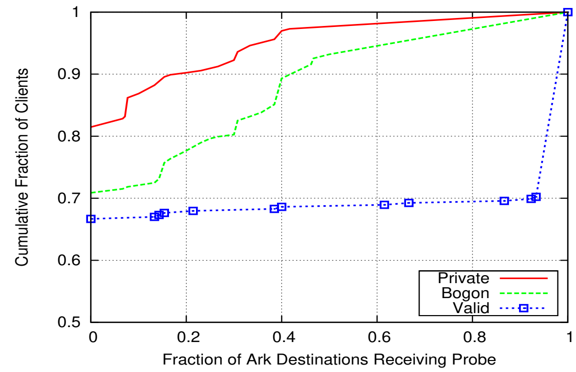


Figure 7: Distribution of client ability to spoof source types to each destination illustrating variation in filtering policies among paths.

that could spoof to zero destinations, i.e. could not spoof to any destination. Approximately 67% of probes with valid, routeable sources, 71% of bogon probes, and 82% of RFC1918 probes are received by none of the Ark destinations. Thus, taking the complement of these percentages, approximately 33% of valid, 29% of bogon, and 18% of private sources reach one or more of our destination receivers. The relatively high incidence rate of clients capable of spoofing traffic with private source addresses is remarkable given that filtering RFC1918 traffic ("martians") involves simple, static, and widely available filtering mechanisms. Thus, a first observation is that a significant fraction of clients can still successfully send a spoofed source address, to at least some destination.

A second observation is the non-trivial variation across each type of spoofed traffic class, indicative of filtering deeper into the network such as shown previously in Figure 4, where a client is most likely able to spoof with a valid address, less so with a bogon, and least likely able to spoof with a private source. These differences correspond directly to our previous discussion (§3.2) on the relative difficulty in implementing each type of best-practice filter.

A natural presumption is that filtering is implemented at the network edge and that filtering policy is consistent without dependence on the destination, in which case a client can either spoof a given source address to zero destinations or to all destinations. Most probes with valid sources do show a strong bimodal distribution: either no destinations or all destinations receive valid spoofed traffic, where only about 4% of clients can spoof to some but not all destinations. However, for bogon and private sources, the dependence on destination is rather striking. Approximately 22% and 16% of the clients could send spoofed probes with bogon and private sources to at least one but not all Ark nodes.

These differences between sources can best be explained in the context of where in the network their corresponding mitigation techniques may be employed. For example, private RFC1918 filters are location independent in the sense that their correctness (blocking traffic with private sources while permitting other sources) does not depend on being placed close to the traffic origination point. Similarly, bogon filtering mechanisms are feasible within the core of the network. Providers adopting such mechanisms, for instance with a

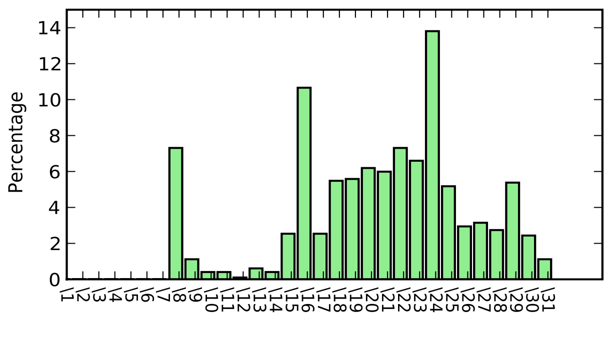


Figure 8: Spoofing neighboring addresses: Probability mass of filtering policy granularity

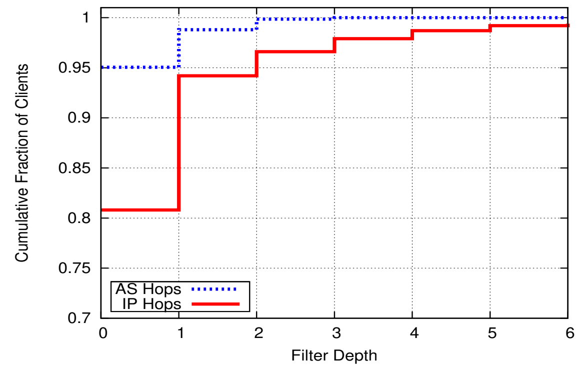


Figure 9: Location of source address validation filters as inferred by the *tracefilter* tool

bogon BGP feed [42], need not continually and manually update filters as new address blocks are allocated. Thus, because of the large spatial diversity achieved by the Ark nodes, our results are evidence of private and bogon filters that are deployed at different points deeper within the network paths.

This is in contrast to available uRPF mechanisms [18, 45, 5] used to prevent forwarding of spoofed traffic with valid sources. uRPF is practical only at the edge: strict mode uRPF creates an intolerable number of false positives deeper in the network while loose mode uRPF is incapable of blocking spoofed traffic when deployed far from the source [39]. As such, there is no effective means for preventing spoofed traffic from reaching the destination if the edge network is incapable or unwilling to perform ingress filtering. Therefore, we see little variation among the number of destinations receiving valid traffic: either ingress filtering is in place near the origin preventing all destinations from receiving probes, or no edge filtering exists. Manual inspection of the small number of clients capable of spoofing valid traffic to an intermediate number of destinations reveals different paths from the origin where one path implements ingress filtering while the other does not, i.e. multi-homed clients with different provider policies.

While methods exist to reliably filter spoofed private and bogon sources, ingress filtering cannot be reliably depended upon to provide widespread spoofing mitigation because of deployment and incentive issues surrounding the available techniques. Thus, our measurements reinforce anecdotal evidence that *concerted spoofing attacks remain a serious concern.*

5.2 Neighbor Spoofing

Next, we examine the prevalence of adjacent neighbor spoofing. At one extreme, a host may be tied to a single IP address, for example using mechanisms that enforce a mapping between a host's link-layer, e.g. Ethernet address and permissible source IP addresses. Cable modems using the DOCSIS specification [14] and DSL modems often enforce these IP bindings.

At the other end of the spectrum, a host may be able to co-opt the addresses of any host on its subnetwork segment. We term the spectrum of neighbor spoofing capability the provider's filtering granularity. Filtering granularity reflects the operational tension between implementing per-host address bindings and maintaining those bindings. In the absence of automated mechanisms [46], network operators and providers have few means to enforce fine-grained anti-spoofing policies.

If the network can rely upon the source information being accurate to the granularity of a service provider, malicious packets can be more easily tracked. Worms and bots, e.g. Wisdom(j), Tsunami, W32.Blaster, etc. now have the capability to spoof addresses within the same subnetwork to evade detection. Data from the adjacent netblock neighbor scan in §3.2 allows us to analyze filtering granularity.

We examine the subset of clients able to spoof none of our three main test source addresses, but verified to be capable of reaching our server with non-spoofed UDP validation traffic. This subset includes 920 clients which probe adjacent neighboring addresses.

Figure 8 displays a probability mass function of implemented ingress filtering granularity as observed in our study. Of interest are the modes at the classful network boundaries of 24 and 16, implying that many networks are internally managed as separate /24 subnetworks. 11% of clients are able to spoof any addresses within the /16 subnetwork to which they belong, implying that these clients may spoof 65,535 other addresses. Over 77% of clients can spoof an address within their own /24 subnetwork. Therefore, *even those clients which cannot spoof arbitrary sources are still frequently afforded anonymity within their own subnetwork.*

5.3 Filter Location

Conventional wisdom dictates that ingress filtering is performed near the network edge rather than the core. Within the core, filter lists become unmanageably large while uRPF techniques may block legitimate traffic due to multi-homing and routing asymmetry. Less restrictive forms of reverse path filtering, loose uRPF [5], allow partial filtering in some cases of asymmetry, but at the cost of unacceptably increased false negatives [39].

However, if anti-spoofing mechanisms are deployed at the edge of the network, the edge contains the largest number of devices and interfaces. Thus appropriately deploying, managing, and maintaining these filters is operationally challenging, particularly when preventing spoofing of valid routeable addresses. As one lower-bound estimate, Bush et al. show

Table 3: Geographic distribution of results

Continent	Spoofing Successes	Spoofing Rate
N. America	498	18.2%
S. America	44	19.4%
Europe	389	19.1%
Asia	289	32.6%
Oceania	40	25.6%
Africa	15	17.4%

Table 4: Client ability to evade filtering based on client AS type relationship to nexthop AS

Client AS to nexthop relationship	Source Address		
	Private	Bogon	Valid
Customer-to-Provider	3.0%	6.5%	25.8%
Sibling	16.2%	40.3%	79.2%
Provider-to-Customer	8.8%	19.9%	35.7%

more than 10% of all autonomous systems filter traffic to and from newly assigned (former bogon) address space [13].

Figure 9 shows the cumulative distribution of provider filter depth, measured in IP and AS hops from the client, as inferred by the tracefilter mechanism described in §3.3. Recall that tracefilter infers the location of filters which block spoofed packets with routeable sources. 80% of the filtered clients are filtered at the first hop IP router. Over 95% of blocked packets are filtered within the same AS as the source. Thus, networks today do generally rely upon the edges to properly validate source information.

These results, in conjunction with anecdotal reports of systemic reachability problems for newly allocated address space, suggest wide distribution of specific edge filters. Edge-centric filtering is not surprising given the limitations of existing filtering mechanisms, however is of significant concern in light of the deployment and incentive issues required of edge filtering. Based on our tracefilter tool and analysis of results from our data set in the next subsection, we infer that: *if spoofed packets successfully traverse the first two network hops, those packets are likely to travel unimpeded to the destination.*

5.4 Origin Network Variation

We assess the geographic distribution of clients in our data set to quantify testing coverage and infer any regional differences. We use the commercial NetAcuity tool [1] to map each client's IP address to a location, shedding light on a different aggregate view of our results. Table 3 numerically divides our results into continental regions, listing the number of unique client tests and the subset that can spoof. While North and South America and Europe see nearly 20% of their clients capable of any spoofing, Oceania is above 25%. Asia stands out with a greater than 32% successful spoofing rate. Although we are as yet unable to completely explain these geographic differences, a plausible explanation is that less developed countries are able to devote fewer resources to network hygiene.

To better understand these geographic variations, we examine the composition of clients based on reverse DNS names and heuristics. Our control server performs reverse DNS lookups asynchronously shortly after a test completes and any missing reverse entries we categorize as "unknown." Figure 10(a) is a stacked histogram of clients on a top-level domain basis, inclusive of the most popular domains. This figure shows the distribution of clients in each domain capable of spoofing. Whereas educational domains are the most well-protected, despite having the lowest NAT rate, Japan and the organizational domains experience greater than 20% successful spoofing rates.

While Figure 10(a) includes only the most frequently observed domains in our data set (those with at least 100 client tests), we perform a series of DNS heuristics [41] to classify each client into network type categories. We observe very low rates of spoofability for cable and DSL networks, likely due to their infrastructural ability, e.g. DOCSIS [14], to bind assigned link-layer addresses to a single IP address. In contrast, clients in commercial networks experience the highest rates of successful spoofing.

Next, we ask whether small or large networks more consistently and effectively implement source address filtering. As a relative network size estimate, we infer each AS's degree based on the number of links to other ASes present in Routeviews. We bin the AS degree into discrete ranges such that each bin is representative of at least several hundred client tests over our entire four-year measurement data set. Figure 11 reports the ability of a client to spoof as a function of that client's origin AS degree.

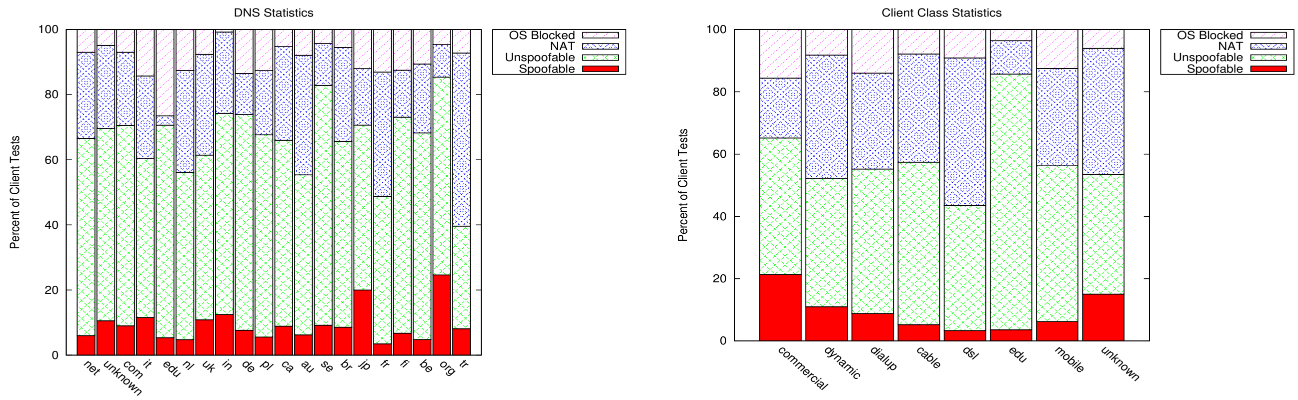
Although one might expect the percent spoofable to increase with AS size, for instance due to difficulty in managing a large and diverse network, no clear trend emerges based on AS degree. In fact, a more plausible explanation is that larger ASes have more resources available to securing their networks as we observe the ability to spoof decrease in relation to AS size up to providers of degree 500. Analysis on our three-month intervals similarly reveal a lack of any correlation to AS degree, although the relative percentages of each discretized bin are larger in the most recent data.

We therefore use a valley-free inference model [19] in conjunction with our collected traceroute data in order to better understand the filtering relationships between providers. Specifically, we wish to determine whether the relationship between the client network's origin AS and the nexthop AS along the path to a given Ark destination correlates with the client's ability to spoof different source addresses. We again map each client to the AS that advertises the route containing the client's IP address. Using the traceroute data from the client to each destination, collected as part of the normal test operation, we determine the true nexthop AS of our spoofed probes along each client-to-destination path.

With the client and nexthop AS pair, we use an inferred valley-free data set [15] to determine one of a provider, customer, sibling, or peer relationship between the ASes. Table 4 shows the relative inter-AS spoofing ability percentages for each type of traffic as a function of these inferred AS relationships[8]

Several points from the AS relationship data bear notice. First, customer to provider links more frequently implement ingress filtering measures and block spoofed traffic.

[8]Too few tests originate from peering networks to form a reliable inference; the union of complete AS paths may provide better data on peer source address validation.



(a) Domain differences in spoofing abilities

(b) Network class differences in spoofing abilities

Figure 10: Observed variation in spoofing abilities across network types and locations

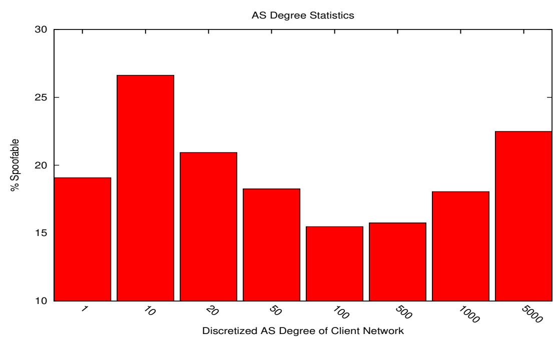


Figure 11: Discretized client origin AS degree versus ability to spoof

However, customer to provider links remain susceptible to spoofed traffic using routeable sources. Sibling relationships understandably show the lowest filtering rates as these relationships are indicative of ASes owned by the same organization. Thus, while private and bogon filtering has been effectively deployed on customer to provider edges, *networks are ill-prepared to provide protection from valid, routeable spoofed sources.*

5.5 Secondary Spoofing Failures

This subsection discusses secondary spoofing failures, where the client is unable to send spoofed source traffic due to local configuration. We entirely omit secondary spoofing failures from our filtering analyses as we cannot disambiguate local effects from provider filtering. However, these secondary effects shed light on an attacker's ability to leverage a compromised machine for the purpose of spoofing.

The most prominent cause of local spoofed packet blocking is NAT. Our server detects instances of NAT in two ways, via the client's locally assigned address and upon reception of a packet whose spoofed source is rewritten. We disregard these sessions, more than 30% of all tests, as we cannot infer whether spoofing would have been successful without the NAT in place.

Recall that clients send non-spoofed UDP packets to establish validity. Approximately 5% of the test instances fail to transmit these ground truth non-spoofed UDP port 53 probes and are ignored in our filtering results. The most likely explanation for these non-spoofed failures is DNS proxies and firewalls. Analysis of the top-level domains of these clients reveals approximately 28% belonging to .net, 20% unknown, 20% .com, 4% .edu, and the remainder in the noise, suggesting that these failed instances do not represent a significant source of measurement bias.

The third cause of local spoofing failures is operating system imposed protection mechanisms. Some operating systems, most notably recent version of Windows, forbid raw sockets from sending spoofed source packets. Similarly, local host firewall rules can present a situation where the raw socket calls succeed, but packets are blocked by the operating system on the send operation. While such operating system-level security enhancements are a positive step toward preventing spoofing which the community should welcome, they are not a complete solution. For example, our testing client detects instances of attempted operating system blocking and adapts by crafting raw Ethernet frames addressed to the host's local router[9].

A compromised host and motivated attacker can circumvent operating system spoofing checks in similar ways, by using raw Ethernet, modifying the operating system, removing firewall rules, etc. Thus, *we must not depend on the security of end-hosts as a solution to source IP address spoofing.*

5.6 Evolution of Filtering

The scope and duration of our data collection, four years and over 12,000 clients, allows us to form initial observations about the evolution of Internet source address filtering. Because network policy and structure will naturally change over such an extended period, we restrict the comparative analysis in this subsection to fixed-size three-month windows in order to mitigate error accumulation while maintaining reasonable and comparable sample sizes.

Since we depend upon voluntary clients running the tester

[9]The OS could further restrict spoofed IP over Ethernet frames, but all OS protections may be circumvented by a determined attacker.

Table 5: Longitudinal comparison between three-month periods in 2005 and 2009 with 1,100 and 400 distinct clients respectively.

Metric	Proportion Spoofable		
	2005 (single dest)	2009 (single dest)	2009 (all dests)
Sessions	18.8±3.2%	29.9±6.0%	31.2±6.0
Netblocks	20.0±3.5%	30.2±6.4%	31.7±6.5
Addresses	5.0±1.8%	11.0±4.1%	11.1±4.1
ASes	23.4±5.0%	31.8±7.6%	34.1±7.6

client, the number of samples received in any fixed time window varies. Some weeks see fewer than 10 clients while other weeks experience more than 1000, corresponding to events where the project is publicly advertised [11]. We therefore examine and compare the two three-month windows with the largest number of tests that are well-separated in time.

The two intervals we examine are separated by four years: a three-month interval surrounding the Slashdot event in 2005 [11] and a recent three-month window starting in February 2009. The two windows include approximately 1,100 and 400 distinct client samples respectively, after removing approximately 35% and 50% of clients from analysis due to secondary effects such as NAT.

Table 5 compares the two periods and provides 95th percentile confidence intervals. As we added the multiple Ark destinations in 2009, we separate the 2009 period into two columns corresponding to the period inclusive of just the single destination and the period with all destinations. In this fashion, we can isolate temporal effects from any effects due to increased resolution from multiple destinations.

We classify a session as "spoofable" if any probe using one of the three main source address types (valid, bogon, or private) reaches any one of the destination Ark receivers. We further require non-spoofed validation traffic to reach the destination in question and the packets must arrive at the destination without having been rewritten by a NAT.

With routing information from RouteViews [28], we map each client to its origin netblock and AS. From the size of an individual client's netblock, we extrapolate the number of IP addresses for which the client's report is representative as a means of providing relative comparison estimates. Note, however, that multiple distinct client reports within one of the three-month windows can provide contradictory information. For example, consider two clients residing on different networks, but belonging to the same AS. If the first client experiences anti-spoofing filters on the local network, while the second client has none, there will be conflicting evidence for at the AS level inference. We resolve such inconsistencies by using the most recently available data.

Unfortunately, not only do we not find any improvement in the prevalence of Internet anti-spoofing filtering, our most recent results reveal an increase in the ability to spoof over the period in 2005. For example, comparing the two time periods where the same, single destination is used (columns 1 and 2), the percent of Sessions that were able to spoof increased from 19% to 30%. The inclusion of multiple destinations (column 3) yields a further, though only minor, increase. We posit that *source validation filtering has not kept up with the Internet's growth and evolution.*

5.7 Spoofing Behind NATs

Finally, to underscore the subtleties in attempting to provide network security through point solutions, we detail a test instance in our results where a NAT did not provide the intended anti-spoofing isolation.

Sending spoofed source packets from behind a NAT is generally assumed to be futile as most NATs rewrite source addresses in order to connect multiple machines behind a single public IP address. Surprisingly, we found a successful tracefilter (§3.3) result for a client despite the client residing behind a source-rewriting NAT. This result was unexpected as tracefilter depends on sending spoofed source packets into the network.

In cooperation with the user, we discovered the following faulty behavior of this particular NAT implementation. The NAT rewrites tracefilter packets with source S to the public external address E while preserving the TTL, as is common in NAT devices. The resulting ICMP messages are returned to the NAT's externally facing interface with address E. The NAT performs the normal reverse mapping and rewrites the ICMP packet to have destination S. However, the faulty implementation does not check on which interface the S to E mapping exists. The NAT determines that these ICMP packets with destination S should be sent out the external interface. Our server, S, successfully receives the spoofed source packets. Thus, the NAT is malfunctioning by maintaining only an S to E mapping and relying on its routing tables to send rewritten packets. Correctly implemented NATs either maintain an $S, E, interface$ tuple, or prevent sources with addresses other than those from the local interface's network. This unexpected implementation illustrates the difficulty in protecting against spoofing. *We cannot reliably anticipate the next exploit or shift in attack patterns that leverage spoofing.*

6. DISCUSSION

Our measurements demonstrate the Internet's vulnerability to IP source address spoofing, and attacks which exploit this vulnerability. While the rise of large-scale zombie farms and NAT deployment can provide a similar level of anonymity, negating the need for spoofing in certain classes of attack, Section 2 demonstrates three recent attacks dependent upon forging IP sources. Recent reverse traceroute methods in the research community even rely on the ability to send spoofed source packets [25], and find networks sufficiently open to permit their methods. The Internet's architectural inability to prevent spoofing implies we cannot reliably anticipate or defend against the next exploit or shift in attack patterns that leverage spoofing. Hence, network operators are forced to rely upon defensive point solutions to mitigate known spoofing attacks. This section examines the larger implications of our results.

6.1 Network Evolution

At the onset of this research [10] we privately predicted that worms and viruses would adaptively and intelligently exploit spoofing as permitted by local conditions. Today we see just such activity and ability (§5.2). A further increase in spoofing is likely given the growing capability of network defenders to quickly respond with selective filters for hosts attacking from fixed addresses. Spoofing provides a means to evade detection by confounding automated systems which attempt to characterize attacks.

As the Internet evolves, new and non-obvious incentives to spoof are arising with potentially far-reaching impact. For example, provider adoption of bandwidth metering or caps in service plans creates an opportunity for strategic users to exhaust the volume budget of their neighbors via spoofing. In general, the confluence of economics, security, and anonymity factors being added to the network create potential incentives to spoof.

IPv6 introduces challenges and opportunities for managing spoofing. Address allocation in IPv6 is intended to alleviate fragmentation and aggregation issues that plague IPv4 networks. Stricter hierarchical address allocation in IPv6 promises to remove global routing table bloat and make many of the issues associated with uRPF filtering manageable. For example, careful IPv6 address assignment enables providers to filter packets from outside their address range without fear of blocking an address from a legitimate downstream customer. However, in practice, many IPv6 networks are obtaining provider independent space, implying that mitigating spoofing in IPv6 networks will be no easier than with IPv4 today. Worse, neighbor spoofing in IPv6 is more problematic as the space of possible neighbor addresses is huge. A provider that assigns an entire /64 to an interface enables malicious hosts to spoof a range of addresses many times larger than the entire 32-bit IPv4 address space.

We recently implemented IPv6 testing in the spoofer client and are just beginning to collect results. Thus far, all IPv6 clients are capable of fully spoofing IPv6 sources, both allocated and unallocated, and even link and site-local sources. This lack of IPv6 filtering suggests that the majority of effort within the IPv6 community is focused on connectivity rather than security. For example, many islands of IPv6 connect through IPv4 tunnels that do not perform IPv6 source validation. As IPv6 becomes more prevalent, we expect spoofing to be at least as problematic as it is in IPv4.

Our measurements indicate that spoofing remains, and will remain, a viable attack vector given the available defense mechanisms. The network is likely to evolve in a direction which makes ingress filtering ever more complex, forcing operators to choose between security and the fragility of their system. Further, *the sophistication of attackers and their uses of source address spoofing will only increase.*

6.2 Defining a Solution Space

Our results provide insights into the Internet's lack of packet-level accountability despite clear need and long historical record of spoofing-based exploits:

- **Protection of provider from spoofed sources:** Providers are driven by economics and have little incentive to deploy, maintain, and support current best practices filtering [3]. Existing spoofed-source filtering mechanisms protect all networks *except* the deploying network who may still receive anonymous, malicious traffic from third-parties that do not properly filter. Therefore, providers realize no benefit, yet must invest resources and risk committing implementation mistakes. Since filtering mistakes can result in accidental blocking of legitimate traffic and customers, e.g. in the event of a routing or configuration change [38], the economic disincentive to filter is strong.

- **Protection of provider's address space:** Providers wish to protect their own address space from being spoofed by hosts within other regions of the network. Existing schemes have no such provision; even with nearly universal deployment of ingress filtering, a provider cannot guarantee that their addresses are not being used maliciously elsewhere in the network.

- **Feasible deployment:** Current best practice filtering is implemented near network edges, and a single unfiltered ingress point provides a means to circumvent global spoofing protection mechanisms. Thus, while ingress filtering is incrementally deployable, the benefit of any single deployment is small. Effective deployment of spoofed source filtering requires large-scale coordinated cooperation, unlikely among competing providers.

- **Speed, scalability, and availability:** uRPF is relatively light weight, but often unavailable or performs poorly even on modern commercial hardware [37].

In response to the relative failure of best practice methods to provide a proactive approach to anti-spoofing, several research initiatives explored the feasibility of reactive techniques. Proposed packet marking [8] and traceback [40] schemes trace spoofed packets to their origin, removing the advantage of anonymity. Unfortunately, while providers have created ad-hoc methods to aid against spoofing-based attacks [20], none of these traceback methods are available in commodity hardware. Thus, reactively finding and blocking spoofed traffic remains a largely manual and time-consuming process, requiring expensive inter-ISP cooperation, rendering it an unsolved problem for network operators.

6.3 Lessons on Anti-Spoofing Design

These observations on existing best practices imply that any voluntary scheme must protect parties who implement it from receiving spoofed traffic without relying on large-scale distributed coordination or cooperation. For a scheme to be viable, it must further protect the deploying provider's own address space from being exploited. Finally, it must be possible to deploy any solution not just at the network edge, but within the network core. Such a solution would permit a small number of large transit providers to significantly affect the ability to spoof.

Anti-spoofing mechanisms are an engineering balance between filtering granularity, accuracy, speed, and cost. Probabilistic schemes that produce false positives [9] should not be ignored: soft responses such as placing suspected spoofed traffic on separate deprioritized queues may be an acceptable design solution given cost and speed constraints.

Our adjacent neighbor spoofing measurements demonstrate that even hosts unable to spoof in general are frequently capable of forging neighboring addresses from within their own network. Thus, there exist natural tradeoffs between the granularity at which source authenticity is enforced, i.e. at the host, network, or AS boundaries, and the cost in providing such protection.

However, where in the network to perform source validation and the appropriate granularity is only part of the architectural design space. Whether to validate the source information based upon implicit or explicit properties of the packet as shown in Figure 12 is an additional consideration. Such a division is a useful metric in evaluating proposals from the research literature:

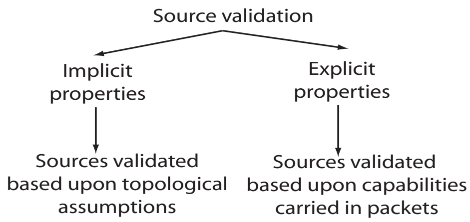


Figure 12: Source validation can be based upon implicit or explicit packet properties, balancing accuracy and validation cost.

- **Explicit:** In an explicit validation scheme, packets or flows carry an explicit authentication token. The authentication may be a cryptographic signature enforced by the end systems or labeling performed by the network as in recent capabilities schemes [26]. Such cryptographic primitives provide much stronger, albeit complex, protection. Alternatively, the StackPi packet marking scheme of Yaar et al. [47] deterministically encodes a packet's path traversal into its IP header. Once a packet is deemed hostile, all subsequent packets with that path identifier can be rejected, regardless of the source address. StackPi does not detect spoofed packets, but permits marking-based filtering, trading accuracy for performance and incremental deployment capability. It also requires adding software support and processing overhead to routers on the Internet and any host interested in the path information, a formidable obstacle to vendor or operational adoption.

- **Implicit:** In an implicit validation scheme, such as the filtering done in today's networks, packets are validated based upon knowledge of local topology and address space assumptions. For example, Jin et al. introduced a scheme to block spoofed packets based on hop count [24], while Duan et al. detail a filtering mechanism based on feasible path construction [16]. Neither of these approaches meets the cost and accuracy requirements of operators.

7. CONCLUSION

Source information in a packet enables two architecturally useful functions: the ability to establish bi-directional communication and meaningful error reporting, e.g. ICMP messages. Unfortunately, IP addresses are overloaded to serve as a network identifier, location, and authentication token. Today, many authentication and authorization policies are based upon IP addresses. A popular means of providing spam protection for example, is dynamic IP blacklists. Such use of source IP addresses for identity may solve an immediate problem, but by its nature is difficult to manage and guarantee correctness – undermining well-meaning attempts at security. Further, the potential damage of unintentionally blacklisting innocent parties may outweigh the protective benefit obtained by IP source-based authentication.

While it is beyond the scope of this measurement work to evaluate the security of clean slate designs, we recognize the fundamental weaknesses of the existing IP architecture that allow IP source address spoofing and prevent any adequate technical response. Clean slate designs that tackle the problem of overloaded IP address semantics, also obviate many problems associated with IP source address spoofing.

Finally, non-technical approaches to mandating and enforcing existing ingress filtering methods represent a second type of clean slate approach, perhaps better considered as "policy architecture" innovation. For instance, governments might enact regulations that require networks to implement filtering, overcoming the aforementioned incentive issues. In such a scenario, networks that facilitate spoofing-based attacks could be held legally liable for damages incurred to victims of such attacks. While Internet regulation is fraught with peril in at least three dimensions of policy architecture (legislative, executive, and judicial), such regulation may be eventually considered an acceptable compromise in return for the security it affords. A softer approach to incentive modification is to provide a weekly public summary of networks observed to permit spoofing and publish the summary to operational mailing lists. Such techniques have been used in the past, for instance the weekly routing table report [6] which identifies providers improperly disaggregating prefixes in the routing table, with success. As with many other aspects of Internet security, a combination of technical and non-technical approaches may be the most tenable solution to mitigating the ability to spoof.

Acknowledgments

The authors would like to recognize Emile Aben, Mike Afergan, Steven Bauer, Simson Garfinkel, Richard Hansen, Aaron Hughes, Simon Leinen, Teemu Schäbl, Karen Sollins, and John Wroclawski for constructive discussions and testing. Tracefilter germinated from an insightful conversation with John Curran. Special thanks to Vern Paxson for invaluable feedback.

8. REFERENCES

[1] Netacuity IP intelligence, 2009. `http://www.digital-element.com/`.
[2] Private communication with Cisco engineering, May 2009.
[3] L. Andersson, E. Davies, and L. Zhang. Report from the IAB workshop on Unwanted Traffic. RFC 4948, Aug. 2007.
[4] Arbor Networks. Worldwide infrastructure security report, 2008. `http://www.arbornetworks.com/report`.
[5] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704, Mar. 2004.
[6] T. Bates, P. Smith, and G. Huston. CIDR Report, 2009. `http://www.cidr-report.org`.
[7] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19:2:32–48, 1989.
[8] S. M. Bellovin. ICMP traceback messages. IETF Internet Draft, Sept. 2000. `http://www.cs.columbia.edu/~smb/papers/draft-bellovin-itrace-00.txt`.
[9] R. Beverly. *Statistical Learning in Network Architecture*. PhD thesis, MIT, June 2008.
[10] R. Beverly and S. Bauer. The Spoofer Project: Inferring the extent of source address filtering on the Internet. In *Proceedings of USENIX SRUTI Workshop*, July 2005.
[11] R. Beverly and S. Bauer. Can you spoof IP addresses? *Slashdot*, May 2006. `http://it.slashdot.org/article.pl?sid=06/05/02/1729257`.
[12] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122, Oct. 1989.
[13] R. Bush, J. Hiebert, O. Maennel, M. Roughan, and S. Uhlig. Diagnosing the location of bogon filters. NANOG 40, June 2007.
[14] Cablelabs. Data over cable service interface specification (DOCSIS), 2006. `http://www.cablemodem.com/`.

[15] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, k. claffy, and G. Riley. AS relationships: inference and validation. *SIGCOMM Comput. Commun. Rev.*, 37(1):29–40, 2007.

[16] Z. Duan, X. Yuan, and J. Chandrashekar. Constructing inter-domain packet filters to control IP spoofing based on BGP updates. In *Proceedings of IEEE INFOCOM*, 2006.

[17] M. Dusi and W. John. Observing routing asymmetry in internet traffic, 2009. `http://www.caida.org/research/traffic-analysis/asymmetry/`.

[18] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000.

[19] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, 2001.

[20] B. R. Greene, C. Morrow, and B. W. Gemberling. ISP security: Real world techniques. NANOG 23, Oct. 2001.

[21] Y. Hyun and k. claffy. Archipelago measurement infrastructure, 2009. `http://www.caida.org/projects/ark/`.

[22] IANA. Special-Use IPv4 Addresses. RFC 3330, Sept. 2002.

[23] V. Jacobsen. Traceroute, 1988. `ftp://ftp.ee.lbl.gov`.

[24] C. Jin, H. Wang, and K. Shin. Hop-count filtering: An effective defense against spoofed DoS traffic. In *Proceedings of the 10th ACM (CCS)*, pages 30–41, Oct. 2003.

[25] E. Katz-Bassett. Practical reverse traceroute. NANOG 45, Jan. 2009.

[26] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: Secure and adoptable source authentication. In *Proceedings of USENIX NSDI*, 2008.

[27] D. Malone and M. Luckie. Analysis of ICMP quotations. In *Proceedings of the 8th Passive and Active Measurement (PAM) Workshop*, Apr. 2007.

[28] D. Meyer. University of Oregon RouteViews, 2007. `http://www.routeviews.org`.

[29] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, 2006.

[30] R. Morris. A Weakness in the 4.2BSD Unix TCP/IP Software. Technical Report 117, AT&T Bell Laboratories, 1985.

[31] C. Morrow. BLS FastAccess internal tech needed, 2006. `http://www.merit.edu/mail.archives/nanog/2006-01/msg00220.html`.

[32] NANOG. DoS attack against DNS?, 2006. `http://www.merit.edu/mail.archives/nanog/2006-01/msg00279.html`.

[33] NANOG. BCP38 business case document, 2007. `http://www.merit.edu/mail.archives/nanog/2007-04/msg00692.html`.

[34] R. Pang, V. Yegneswaran, P. Barford, and V. Paxson. Characteristics of Internet Background Radiation. In *Proceedings of ACM Internet Measurement Conference*, Oct. 2004.

[35] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *Computer Communications Review*, 31(3), July 2001.

[36] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918, Feb. 1996.

[37] J. Rhett. Force10 gear, 2008. `http://mailman.nanog.org/pipermail/nanog/2008-September/003524.html`.

[38] P. Savola. An effect of ignoring BCP38, 2008. `http://mailman.nanog.org/pipermail/nanog/2008-September/003758.html`.

[39] P. Savola. Experiences from Using Unicast RPF. IETF Internet Draft, Jan. 2008. `http://tools.ietf.org/id/draft-savola-bcp84-urpf-experiences-03.txt`.

[40] A. C. Snoeren, C. Partridge, L. A. Sancheq, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. In *Proceedings of ACM SIGCOMM*, 2001.

[41] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networks*, 12(1):2–16, 2004.

[42] R. Thomas. Team Cymru bogon route-server project. `http://www.cymru.com/`.

[43] J. Touch. Defending TCP Against Spoofing Attacks. RFC 4953, July 2007.

[44] US-CERT. Multiple DNS implementations vulnerable to cache poisoning VU#800113, 2008.

[45] P. Vixie. Securing the edge, Oct. 2002. `http://www.icann.org/en/committees/security/sac004.txt`.

[46] C. Vogt. A solution space analysis for first-hop ip source address validation. IETF Internet Draft, Jan. 2009. `http://www.ietf.org/internet-drafts/draft-ietf-savi-rationale-00.txt`.

[47] A. Yaar, A. Perrig, and D. Song. StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense. *IEEE Selected Areas in Communications*, Oct. 2006.

APPENDIX

To remove any potential ambiguity in describing the various analyses in this paper, we specify the following notation:

Define T as the set of n tests $t_1, \ldots, t_n$. Let ϕ^x be a probe of type $x \in \{p, b, v\}$ (private, bogon, or valid source) where $p = \texttt{172.16.1.100}, b = \texttt{1.2.3.4}, v = \texttt{6.1.2.3}$ (Table 1). Neighboring sources in §5.2 are not considered part of the valid probing. Let ρ be a non-spoofed validation probe. Let $D(t_i)$ be the set of destinations for a given test t_i. Let $\phi_i^x \leadsto d$ indicate that the probe of type x for the i'th test successfully reached destination d. Let $\phi_i^x \bowtie d$ indicate that the probe was blocked by either a NAT or the operating system. Define probe indicator variables:

$$v_i^d = \begin{cases} 1 \text{ if } \rho_i^x \leadsto d \\ 0 \text{ otherwise} \end{cases} \qquad t_i^x = \begin{cases} 1 \text{ if } \exists d \in D(t_i) \text{ s.t. } \phi_i^x \leadsto d \wedge v_i^d \\ \emptyset \text{ if } \exists d \in D(t_i) \text{ s.t. } \phi_i^x \bowtie d \\ 0 \text{ otherwise} \end{cases}$$

$$\begin{aligned} \textit{spoofany}(T) &:= \{t_i \in T : \forall x\ \exists t_i^x = 1\} \\ \textit{spoofvalid}(T) &:= \{t_i \in T : \exists t_i^v = 1\} \\ \textit{spoofnone}(T) &:= \{t_i \in T : \forall x\ \nexists t_i^x = 1\} \\ \textit{clientblock}(T) &:= \{t_i \in T : \forall x\ \exists t_i^x = \emptyset\} \end{aligned}$$

Define $T' = \textit{spoofany}(T) + \textit{spoofnone}(T) - \textit{clientblock}(T)$ as a convenient notation for the subset of T clients that are valid and not blocked by secondary filtering. Table 6 details the data set used in each of the paper's experiments.

Table 6: Per-experiment data sets analyzed

Analysis	Data Set	Period
§5.1	T'	Feb 09 – Apr 09
§5.2	$\textit{spoofnone}(T')$	Feb 05 – Apr 09
§5.3	$T' - \textit{spoofvalid}(T')$	Feb 05 – Apr 09
Fig. 10	$T' + \textit{clientblock}(T)$	Feb 05 – Apr 09
§5.4	T'	Feb 05 – Apr 09
§5.5	$\textit{clientblock}(T)$	Feb 05 – Apr 09
§5.6	T'	Feb-Apr 05; Feb-Apr 09