Using Loops Observed in Traceroute to Infer the Ability to Spoof

Qasim Lone[1], Matthew Luckie[2], Maciej Korczyński[1], and Michel van Eeten[1]

[1] Delft University of Technology, the Netherlands
Q.B.Lone, Maciej.Korczynski, M.J.G.vanEeten@tudelft.nl
[2] University of Waikato, New Zealand
mjl@wand.net.nz

Abstract. Despite source IP address spoofing being a known vulnerability for at least 25 years, and despite many efforts to shed light on the problem, spoofing remains a popular attack method for redirection, amplification, and anonymity. To defeat these attacks requires operators to ensure their networks filter packets with spoofed source IP addresses, known as source address validation (SAV), best deployed at the edge of the network where traffic originates. In this paper, we present a new method using routing loops appearing in traceroute data to infer inadequate SAV at the transit provider edge, where a provider does not filter traffic that should not have come from the customer. Our method does not require a vantage point within the customer network. We present and validate an algorithm that identifies at Internet scale which loops imply a lack of ingress filtering by providers. We found 703 provider ASes that do not implement ingress filtering on at least one of their links for 1,780 customer ASes. Most of these observations are unique compared to the existing methods of the Spoofer and Open Resolver projects. By increasing the visibility of the networks that allow spoofing, we aim to strengthen the incentives for the adoption of SAV.

1 Introduction

Despite source IP address spoofing being a known vulnerability for at least 25 years [6], and despite many efforts to shed light on the problem (e.g. [7,8,9]), spoofing remains a viable attack method for redirection, amplification, and anonymity, as evidenced in February 2014 during a 400 Gbps DDoS attack against Cloudfare [19]. That particular attack used an amplification vector in some implementations of NTP [19]; a previous attack against Spamhaus [10] in March 2013 achieved 300+ Gbps using an amplification vector in DNS. While some application-layer patches can mitigate these attacks [20], attackers continuously search for new vectors.

Defeating amplification attacks, and other threats based on IP spoofing, requires providers to filter incoming packets with spoofed source IP addresses [11] – in other words, to implement BCP 38, a Best Current Practice also known as source address validation (SAV). SAV suffers from misaligned incentives: a network that adopts SAV incurs the cost of deployment, while the security benefits diffuse to all other networks. That being said, SAV is a widely supported norm in the community. Increasing the visibility of which networks have or have not adopted SAV reduces the incentive problem by leveraging reputation effects and the pressure of other providers and stakeholders. These factors put a premium on our ability to measure SAV adoption.

In this paper, we report on the efficacy of a new measurement technique that is based on an idea of Jared Mauch. It allows an external observer to use traceroute to infer the absence of filtering by a provider AS at a provider-customer interconnect. This study makes the following five contributions: (1) We show that it is generally feasible for providers to deploy static ingress ACLs, as their customers rarely change address space. (2) We describe a scalable algorithm for accurately inferring the absence of ingress filtering from specific patterns in traceroute data. (3) We validate the algorithm's correctness using ground truth from 7 network operators. (4) We demonstrate the utility of the algorithm by analyzing Internet-scale inferences we made. (5) We build a public website showing the provider-customer edges that we inferred to imply the absence of filtering, combined with actionable data that operators can use to deploy filtering.

2 Background on Ingress Filtering

The canonical documents describing the use of ingress filtering methods for SAV are RFCs 2827 [11] and 3704 [5], known in the network operations and research communities as BCPs 38 and 84. BCP 38 describes the basic idea: the source address of packets should be checked at the periphery of the Internet against a set of permitted addresses. For an access network, this check could be at the point of interconnection with a single customer; for an enterprise, this could be on their edge routers to their neighbors; and for a transit provider, this could be on the provider-edge router where a customer connects. For single-homed customers, a transit provider can discard packets that have a source address outside the set of prefixes the customer announces to the transit provider, using Strict or Feasible Reverse Path Forwarding (RPF). A router using Strict RPF will drop a packet if it arrived on a different interface than the router would choose when forwarding a packet to the packet's source address; a router using Feasible RPF will consider all paths it could use to reach the source address, not just the best path.

BCP 84 discusses challenges in deploying ingress filtering on multi-homed networks. Both Strict and Feasible RPF are not always feasible if a customer is multi-homed and does not announce all of its prefixes to each neighbor router, as it might do for traffic engineering purposes. Instead, an operator might define a set of prefixes covering source addresses in packets the router will forward, known as an Ingress Access List, or Ingress ACL. BCP 84 states that while ingress ACLs require manual maintenance if a neighbor acquires additional address space, they are "the most bulletproof solution when done properly", and the "best fit ... when the configuration is not too dynamic, .. if the number of used prefixes is low."

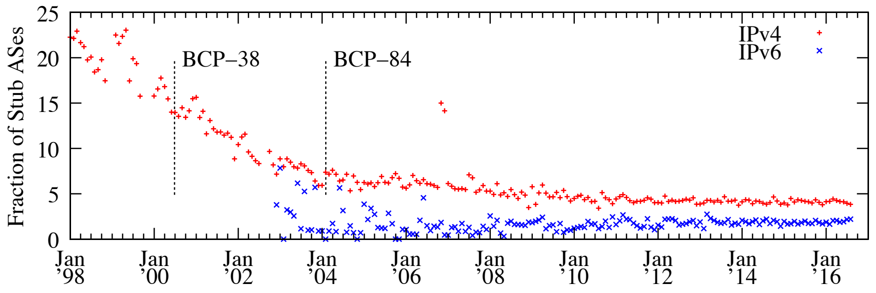


Fig. 1: Fraction of ASes whose prefix announcements changed month-to-month.

3 Related Work

Testing a network's SAV compliance requires a measurement vantage point inside (or adjacent to) the network, because the origin network of arbitrary spoofed packets cannot be determined [5]. The approach of the Spoofer project [7] is to allow volunteers to test their network's SAV compliance with a custom client-server system, where the client sends spoofed packets in coordination with the server, and the server infers that the client can spoof if the server receives these spoofed packets. However, the Spoofer project requires volunteer support to run the client to obtain a view from a given network. In May 2016, CAIDA released an updated client [1] that operates in the background, automatically testing attached networks once per week, and whenever the system attaches to a network it has not tested in the previous week. The number of prefixes tested per month has increased from ≈ 400 in May 2016 to ≈ 6000 in December 2016 [1].

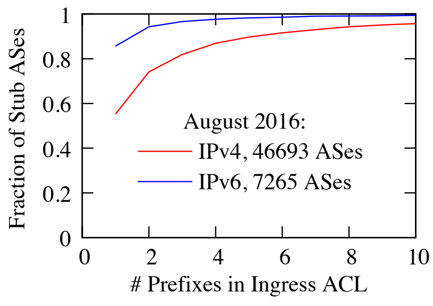
Jared Mauch deployed the first technique to infer if a network had inadequate SAV without requiring a custom client-server system. As a product of the Open Resolver Project [3], he observed DNS resolvers embedded in home routers forwarding DNS queries from his system with IP_X to other resolvers, without rewriting the source IP address of the packet. These other resolvers returned the subsequent answer directly to IP_X, rather than to the DNS resolver in the home router as they should have.

We emphasize that these methods are complementary, and that no one technique is able to test deployment of SAV for all networks.

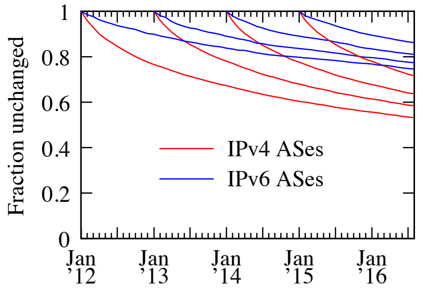
4 Motivation of Ingress ACLs

As described in §2, the best place to deploy filtering is at the edge. However, not all edge networks have the technical ability or motivation to filter their own traffic. A transit provider, however, is often managed by skilled network operators who may already deploy defenses to prevent their customers from announcing inappropriate routes. The provider-customer interconnect for an edge network represents the other straightforward place to deploy ingress filtering.

Figure 1 quantifies the dynamism of address space announced by stub ASes over time. Using BGP data collected by Routeviews and RIPE RIS with the method described in §5.1, we aggregated the prefixes each stub AS originated



(a) Size of Ingress ACLs



(b) Dynamism of Ingress ACLs

Fig. 2: Size and dynamism of ACLs to filter traffic from stub ASes.

in BGP into the minimum prefix set, and examined month-to-month changes in the set. Perhaps a consequence of IPv4 address exhaustion, we see a trend toward stable announcement patterns. This trend may improve the practicality of static ingress ACLs: in May 2000, ≈ 15% of stub ASes would have required deployment of a different IPv4 ingress ACL month-to-month, but in 2015, less than 5% of ASes would have required the same.

As BCP 84 states that because ingress ACLs require manual maintenance they are best suited "when the configuration is not too dynamic" and "if the number of used prefixes is low", figure 2 examines the size and dynamism of ingress ACLs required for stub ASes in August 2016. Figure 2a shows that 88.9% of stub ASes would require an IPv4 ACL of no more than 4 prefixes, and 85.6% of stub ASes would require an IPv6 ACL of a single prefix. Figure 2b shows the dynamism of these ACLs over time, based on ACLs that could have been defined for all stub ASes in January 2012, 2013, 2014, and 2015. For stub ASes for these times, at least 77.4% of IPv4 ACLs would not have had to change over the course of one year; for those defined in January 2012, 54.4% of the inferred ACLs would not have required change even up to August 2016. Further, required IPv6 ACLs would be even less dynamic: more than 74.6% of IPv6 ACLs would not have needed to change over the course of 4.5 years until August 2016. We believe the observed number of prefixes and dynamism over time imply that ingress ACLs are feasible in the modern Internet.

5 Inferring Absence of Ingress Filtering using Traceroute

The key idea of our approach is that traceroute can show absence of ingress filtering by providers of stub ASes when a traceroute path reaches the stub AS and then exits out of the stub, as the traceroute packets contain a source address belonging to the vantage point (VP) launching the traceroute. If the provider's border router is performing SAV, it should filter the traceroute packet when it arrives from the stub AS, as the packet has a source address not belonging to the stub AS. If the provider's router does not perform SAV, it will forward the packet, and the traceroute will show an apparent IP-level forwarding loop as the provider's router returns subsequent packets to the stub AS.

Xia *et al.* found that 50% of persistent loops were caused by a border router missing a "pull-up route" covering address space not internally routed by the customer [21]. However, a forwarding loop does not imply absence of SAV at the edge: a loop resulting from a transient misconfiguration or routing update can occur anywhere in the network. The key challenge in this work is inferring the provider-customer boundary in traceroute [16,18]. In this paper, we superimpose millions of traceroutes towards random IP addresses in /24 prefixes to build a topology graph, and use a small set of heuristics to infer provider-customer edges for stub ASes in the graph. §5.1 describes the Internet topology datasets that we used, and §5.3 describes the algorithm we used to filter the loops that imply the absence of ingress filtering by the provider – in other words, the lack of compliance with BCP 38.

5.1 Input Data

CAIDA IPv4 routed /24 topology datasets: We used CAIDA's ongoing traceroute measurements towards every routed /24 prefix in the Internet. CAIDA's probing of all routed /24s is especially useful here, as the goal is to find unrouted space that can result in a forwarding loop. CAIDA's traceroute data is collected with scamper [15] using Paris traceroute which avoids spurious loops by keeping the ICMP checksum value the same for any given traceroute [4]. As of August 2016, CAIDA probes every routed /24 using 138 Vantage Points (VPs) organized into three teams; each team probes the address space independently. Each team takes roughly 1.5 days to probe every routed /24.
CAIDA IPv4 AS relationships: We used CAIDA's ongoing BGP-based AS relationship inferences [17] to identify customer-provider interconnections in traceroute paths. The relationship files were inferred by CAIDA using public BGP data collected by Routeviews and RIPE RIS, using RIB files recorded on the 1-5 of each month. We also used the same BGP data to identify the origin AS announcing each prefix measured with traceroute.
CAIDA Sibling Inferences: We used CAIDA's ongoing WHOIS-based AS-to-organization inference file [13] to identify ASes that belong to the same underlying organization (are siblings). The sibling files were inferred by CAIDA using textual analysis on WHOIS databases obtained from Regional Internet Registries (RIRs) at 3-month intervals. We used sibling inferences to avoid misclassifying a loop that occurs within a single organization using multiple ASes as one that occurs between distinct provider and customer ASes.

5.2 Construction of Topology

Our first goal is to correctly identify the provider-customer boundaries towards stub ASes with high precision. Because the customer usually uses address space provided by the provider to number their interface on their router involved in the interconnection, the customer-edge router usually appears in traceroute using an IP address routed by the provider. Therefore, one of our goals is to accurately identify customer routers using provider address space without incorrectly inferring that a provider's backbone router belongs to a customer.

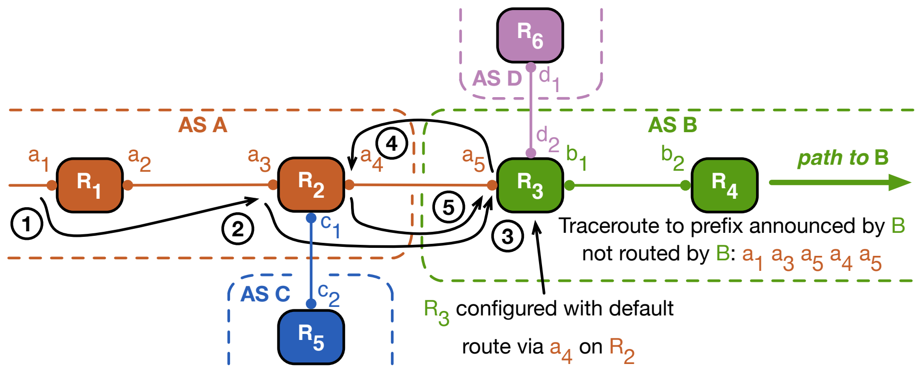

R6
AS D
d1
AS A
AS B
a1
R1
a2
a3
R2
a4
a5
R3
d2
b1
b2
R4
path to B
c1
AS C
c2
R5
1
2
3
4
5
Traceroute to prefix announced by B
not routed by B: a1 a3 a5 a4 a5
R3 configured with default
route via a4 on R2

Fig. 3: A simple loop between AS A and its customer B implying absence of filtering by A at R_2. R_2 should discard packet 4 because it arrives with a source address outside of B's network, rather than send it back to B (5).

We assemble all traceroutes collected for a single cycle by a single team that do not contain loops, and label each interface with (1) the origin AS of the longest matching prefix for the interface address, and (2) the set of destination ASes the interface address is in the path towards. If an address is in the path towards multiple ASes, the address could not be configured on a customer router of a stub AS.

5.3 Algorithm to Infer Absence of Ingress Filtering from Loops

Our algorithm considers two different ways a traceroute path may enter a stub AS and exit through a provider AS: (1) a simple point-to-point loop between a single provider-edge router and a single customer-edge router, (2) a loop from a customer-edge router that exits using a different provider.

Simple point-to-point loops: Figure 3 illustrates the first case, where R_3 is a customer-edge router belonging to AS B configured with a default route via R_2. If the operator of B announces address space in BGP but does not have an internal route for a portion of that address space, and does not have a "pull-up route" covering the unused portion on R_3, then R_3 forwards the packet back to R_2 using the default route [21]. R_2 will then forward the packet back to R_3, the loop sequence will likely be a_5 (customer-edge router), a_4 (provider-edge router), and a_5 (customer-edge router), with a_4 and a_5 assigned from the same IPv4 /30 or /31 prefix the routers use to form the point-to-point link. Therefore, our criteria are: (1) that the addresses in the loop are assigned from a single /30 or /31 prefix, (2) that the AS originating the longest matching prefix is an inferred provider of the stub AS and not a sibling of the stub AS, (3) that the assumed customer router only appears in traceroute paths towards the stub AS, (4) that there is at least one other address originated by the provider in the traceroute path towards the stub. Criteria #3 avoids incorrectly inferring a provider-operated router as a customer-edge router when a loop occurs before the stub AS (e.g. a_1 a_3 a_2 a_3) as a_3 appears in traceroute paths towards both

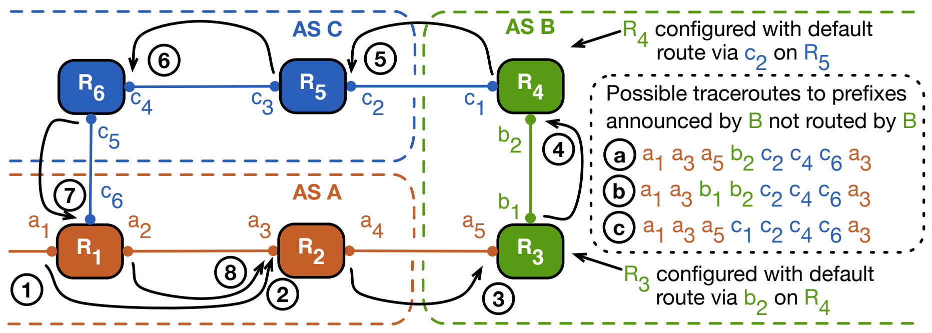

AS C
AS B
AS A
R4 configured with default route via c2 on R5
R3 configured with default route via b2 on R4
Possible traceroutes to prefixes announced by B not routed by B
a a1 a3 a5 b2 c2 c4 c6 a3
b a1 a3 b1 b2 c2 c4 c6 a3
c a1 a3 a5 c1 c2 c4 c6 a3
R6
R5
R4
R1
R2
R3
c4
c3
c2
c1
c5
c6
b2
b1
a1
a2
a3
a4
a5
1
2
3
4
5
6
7
8

Fig. 4: A two-provider loop between ASes A and C and their customer B implying absence of filtering by C at R_5. R_5 should discard packet 5 because it arrives with a source address outside of B, rather than forward the packet to R_6.

B and C. Criteria #4 avoids incorrectly inferring which router in a traceroute path is the customer-edge router when the customer-edge router is multi-homed and the traceroute path enters via a second provider AS D (e.g., d_2 a_4 a_5 a_4).

Two-provider loops: Figure 4 illustrates the second case, where R_3 and R_4 are customer-edge routers belonging to AS B, with default routes configured on R_3 and R_4. The underlying routing configuration issues are the same as a point-to-point loop, except the default route is via a different AS than the AS the traceroute entered the network. Figure 4 shows the traceroute visiting two routers operated by AS B; however, it is possible that the traceroute will never contain an IP address mapped to B, depending on how many routers in B the traceroute visits, and how the routers respond to traceroute probes. Therefore, our criteria are: (1) that the assumed customer router where the traceroute exits appears only in paths towards the stub AS, (2) that both the ingress and egress AS in the traceroute path are inferred providers of the stub AS and not a sibling of the stub AS, (3) that there is no unresponsive traceroute hop in the traceroute path where a customer router could be located, (4) that at least two consecutive IP addresses mapped to the same egress AS appear in the loop. Criteria #2 does not require different provider ASes: if the stub AS is multi-homed to the same provider with different routers, our method will still infer an absence of filtering. Criteria #3 ensures that we do not mis-infer where the customer router is located in the path, and thus incorrectly infer the AS that has not deployed ingress filtering. Finally, criteria #4 reduces the chance that a loop inside the customer network is mis-classified as crossing into a provider network if the customer router responds with a third-party IP address.

5.4 Finding Needles in a Haystack

As discussed in §5.1, CAIDA uses three teams of Ark VPs to probe a random address in every routed /24 prefix. In this section, we report on the characteristics of cycle 4947 conducted by team 3. The characteristics of data conducted by other teams and for other cycles is quantitatively similar. In total, cycle 4947 contains 10,711,132 traceroutes, and 163,916 (1.5%) of these contain a loop. 105,685 (64.5%) of the traceroutes with loops were not towards a stub network.

Of the remaining 58,231 traceroutes with loops towards stub ASes, we inferred 31,023 (53.3%) had a loop within the stub network, i.e. the addresses in the loop were announced in BGP by the stub, or involved the customer-edge router. A further 11,352 traceroutes (19.5%) contained a loop with an unresponsive IP address, and 1,373 traceroutes (2.4%) contained an unrouted IP address that prevented us from inferring if the loop occurred at a provider-customer interconnect. 610 traceroutes (1.0%) had a loop that we disqualified as occurring at a customer-provider boundary, as the loop occurred at a router that also appeared in paths towards multiple destination ASes, and 494 traceroutes (0.8%) contained an IP address that could have been a third party address on a customer router, rather than a router operated by a provider. In total, only 2,530 traceroutes with loops (4.3%) contained simple point-to-point loops, and only 93 (0.2%) contained more complex two-provider loops.

5.5 Persistence of Loops

Given that we are looking for needles in haystacks, how reliably can we find them? Ideally, we would be able to consistently reproduce the loops that imply absence of ingress filtering, and discard observations caused by transient events. Unfortunately, there is currently no straightforward way of doing so.

The data we used was collected by CAIDA using traceroutes conducted by a distributed set of VPs towards a random IP address in each routed /24 prefix. This approach adds efficiency by reducing the number of probes, at the cost of potentially missing loops that occur for smaller prefixes. It also means that when such a loop is in fact discovered, the next probe might miss it again by selecting a random address outside the smaller prefix. In other words, the traceroute data itself does not tell us much about the persistence of loops.

To better understand the impact of random address selection and the persistence of loops, we collected traceroutes towards the same addresses that revealed the loops. We first applied the algorithm outlined in §5.3 to the traceroute data for August 2016 and found 2,500 unique loops between 703 provider and 1,780 customer ASes. In October 2016, we collected traceroutes towards the same IP addresses that revealed the loops, using two different vantage points. We were able to reproduce 1,240 of the loops between 461 provider and 1,026 customer ASes. Next, we repeated this procedure for over a year of traceroute data: August 2015-August 2016. We found 7,784 unique loops between 1,286 provider and 3,993 customer ASes. In October 2016, we were able to reproduce 1,542 unique loops between 505 provider and 1,176 customer ASes. In other words, the additional data identified 342 loops that persisted.

A significant portion of all loops could not be reproduced and the longer the time lag, the higher the odds of failure, for four reasons. First, the loop might have been transient, i.e., it only occurred during routing protocol convergence [12] or temporary misconfiguration [21]. Second, it might depend on the vantage point of the probe, e.g., because of multi-homed routers. Third, the provider might have fixed the routing issue that caused the loop. Fourth, and most relevant, the provider has implemented ingress filtering.

Future work is needed to untangle these causes. We know from our validation effort (§6) that even loops that appeared only once can correctly signal absence of ingress filtering. Some of the loops that we could not reproduce had already been validated by the provider as true positives. In the remainder of the paper, we will work with the full set of loops as identified by our algorithm.

6 Validation by Network Providers

In order to validate our results and obtain ground truth, we contacted providers in two rounds: September 2015 and September 2016. We got feedback from one hosting provider, four ISPs, two national research and education networks, and two Tier 1 networks. We contacted some providers only in one round, some in both, depending on whether we inferred absence of ingress filtering for links involving their network at both times, and our ability to reach the right specialist in the organization. We gave all providers a formal assurance that their names would not be included in the paper.

Feedback from the providers during the first round resulted in improvements in our methodology. We applied the final methodology to both the August 2015 and August 2016 data. We then compared the final results to the feedback that we received from the providers in both rounds. We talked to 6 providers in round 1 and 7 in round 2, and 4 providers participated in both rounds.

We defined a result as a true positive if we identified a provider-to-customer link where the provider does not perform ingress filtering and an operator at the provider confirms this. That is, we correctly inferred the absence of SAV as well as the boundary between provider and customer. A false positive occurred when we either incorrectly detected the boundary or the provider is actually performing SAV at the boundary. Our methodology correctly identified the absence of ingress filtering on the provider boundary in 95 out of 97 IP links between provider and customer ASes (45 of 47 links in round 1, and 50 of 50 links in round 2).

The two false positives had different causes. One of them occurred because of route aggregation. Providers perform route aggregation by consolidating multiple routes in a single, more general route. This practice can lead to problems with our border router detection. Imagine this scenario: a provider is assigned a /16 prefix X by the Regional Internet Registry (RIR). The provider allocates a /24 subnet Y from prefix X to a customer, and the customer assigns addresses from Y to its routers. The customer also has its own prefix Z allocated by an RIR. If the provider aggregates Y into a single /16 advertisement for X, we would infer that customer routers with addresses in Y belong to the provider AS. Our methodology would then categorize a loop between provider prefix X and customer prefix Z as signaling the absence of SAV, when the loop was actually within the customer network.

For the second false positive, the provider informed us that the traceroute data suggested that the loop had occurred inside their network rather than on the boundary. However, they could not reproduce it anymore and blamed it on a transient event. Note that in the second round, we found 2 loops for the same provider and they were both true positives.

One additional piece of feedback that we received was that some of the providers, while confirming the validity of our inference that they were not doing ingress filtering on their boundary, objected to the implication that they *should* be filtering. They saw their services as offering transit and contracted them as such, which meant no filtering on the provider's side. In the view of these providers, the downstream customer AS should perform SAV at their border router. The customer ASes were business entities like ISPs, hosting providers or large enterprises. Evaluating whether this interpretation of BCP 38 [11] is merited falls outside the scope of this paper and is for the community to address. For this paper, the key point is that the proposed method performed accurately.

7 Results

We first summarize the results in terms of the number of networks that do not implement SAV. We then compare our method to the two alternatives: the Spoofer and the Open Resolver projects. Like those methods, our approach only observes a subset of the networks without SAV. In the absence of loops, we cannot tell anything about the presence of ingress filtering.

Using one month of CAIDA's traceroute data from August 2016, our approach identified 2,500 unique loops involving 703 provider ASes as lacking SAV on one or more of their customer-facing links and 1,780 customer ASes. These represent approximately 1.3% and 3.2% of all advertised ASes, respectively. Moreover, when compared to all advertised stub ASes and their providers [17], we found 9.0% of provider ASes without ingress filtering involving 3.8% of all stub ASes.

As discussed in §6, some providers argued that customer ASes should be responsible for SAV within their networks or at their borders. However, we found that about 63% of the involved customer ASes advertise /20 or smaller prefix lengths. It is unlikely that such small entities have the resources and incentives to implement SAV in their networks. On the other hand, such small prefixes should allow the providers to implement static ACLs.

We now compare our results to the data from the Spoofer and Open Resolver projects (see §3 for details). Our method only detects the lack of ingress filtering for provider networks, which means that their customer ASes might be able to spoof. We compared those customer ASes with the Spoofer data from February to August 2016 [1]. Of 54 overlapping ASes, 38 of the Spoofer tests were only conducted from behind a Network Address Translation (NAT) device that likely prevented spoofing. Of the systems not behind a NAT, 10 of the 16 stub ASes allowed spoofing, i.e., more than half of these ASes had not deployed SAV, suggesting the provider's expectation for their customers to deploy filtering is not being met, and supporting the case for transit providers to filter their customers. This means that the connected provider ASes do not implement ingress filtering, which is consistent with our results. Packets with spoofed source addresses from Spoofer tests in the 6 remaining customer ASes were not received, suggesting that filtering took place in the customer AS. The overlap between both methods contains only a small sample, but it does indicate that the majority of the overlapping customer networks were not doing SAV – a finding that reinforces the point that providers should not expect their customer ASes to be willing and able do SAV, even if they are not that small.

Kührer *et al.* used the Open Resolver data in 2014 by to identify 2,692 unique ASes from within which spoofing was possible [14]. Following the same approach, we analyzed the August 2016 data from the Open Resolver project, generously provided to us by Jared Mauch, and found a total of 3,015 unique ASes that were able to spoof. We compared these to the customer ASes that our method identified as allowing spoofing – i.e., those connected to the providers which lack ingress filtering. We found only a modest overlap: 244 ASes.

In sum: these findings show that our method can add unique data points to both existing methods, and improve visibility of networks lacking SAV. In terms of the volume of observations, it resides between Spoofer and Open Resolver. The three methods are complementary and provide views into the problem, contributing to improved overall visibility of SAV adoption.

8 Conclusion

In this paper we implemented and validated an algorithm that uses traceroute data to infer a lack of SAV between a stub and provider network. We inferred 703 providers that do not implement ingress filtering on at least one of their links facing 1,780 customer ASes. We also built a public website showing the provider-customer edges that we inferred as lacking ingress filtering: `https://spoofer.caida.org/`. Providers can use the data to deploy filtering, which would not only stop attackers from sending packets with spoofed addresses from the customer's network, but also block attempts to attack the provider-customer link by sending packets to addresses that enter the forwarding loop [21].

To improve the reliability of the method, future work is needed on border detection and on untangling the different factors that prevent loops from being reproduced, to separate the implementation of ingress filtering from the other causes. A completely different direction for future work is to experimentally test the strength of reputation effects among providers and network operators. The networks that allow spoofing could be made public in varying ways, to see which mechanism best incentivizes providers into taking action.

For the community of network operators, the results support efforts such as the Routing Resilience Manifesto [2] and other community initiatives to improve network security. By complementing the Spoofer and Open Resolver data, our method increases visibility into the adoption of SAV. Public visibility of spoofing-enabled networks is a critical step in incentivizing providers to deploy ingress filtering in their networks. The dataset is also useful for the national CERTs who want to push BCP 38 compliance in their countries. The problems caused by IP spoofing have been recognized for years [6], and the task to reduce its role in attacks is becoming increasingly urgent.

Acknowledgments: The technique in this paper is based on an idea from Jared Mauch. Christian Keil (DFN-CERT) provided informative feedback. This work was partly funded by the EU Advanced Cyber Defence Centre (ACDC) project CIP-ICT-PSP.2012.5.1 #325188. This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency, Cyber Security Division BAA HSHQDC-14-R-B0005, and the Government of United Kingdom of Great Britain and Northern Ireland via contract number D15PC00188.

References

1. CAIDA spoofer project. `https://spoofer.caida.org/`
2. Mutually Agreed Norms for Routing Security (MANRS), `https://www.routingmanifesto.org/manrs/`
3. Open Resolver Project, `http://openresolverproject.org/`
4. Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., Teixeira, R.: Avoiding traceroute anomalies with Paris traceroute. In: IMC. pp. 153–158 (Oct 2006)
5. Baker, F., Savola, P.: Ingress filtering for multihomed networks. RFC 3704 (Mar 2004), IETF BCP84
6. Bellovin, S.: Security problems in the TCP/IP protocol suite. CCR 19(2), 1989
7. Beverly, R., Bauer, S.: The spoofer project: Inferring the extent of source address filtering on the Internet. In: Proceedings of USENIX SRUTI (Jul 2005)
8. Beverly, R., Berger, A., Hyun, Y., k claffy: Understanding the efficacy of deployed Internet source address validation. In: IMC. pp. 356–369 (Nov 2009)
9. Beverly, R., Koga, R., kc claffy: Initial longitudinal analysis of IP source spoofing capability on the Internet (Jul 2013), `http://www.internetsociety.org/`
10. Bright, P.: Spamhaus DDoS grows to Internet-threatening size (Mar 2013)
11. Ferguson, P., Senie, D.: Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827 (May 2000), IETF BCP38
12. Francois, P., Bonaventure, O.: Avoiding transient loops during IGP convergence in IP networks. In: INFOCOM. pp. 237–247 (Mar 2005)
13. Huffaker, B., Keys, K., Koga, R., kc claffy: CAIDA inferred AS to organization mapping dataset, `https://www.caida.org/data/as-organizations/`
14. Kührer, M., Hupperich, T., Rossow, C., Holz, T.: Exit from Hell? Reducing the impact of amplification DDoS attacks. In: USENIX Security (Aug 2014)
15. Luckie, M.: Scamper: a scalable and extensible packet prober for active measurement of the Internet. In: IMC. pp. 239–245 (Nov 2010)
16. Luckie, M., Dhamdhere, A., Huffaker, B., Clark, D., k claffy: bdrmap: Inference of borders between IP networks. In: IMC. pp. 381–396 (Nov 2016)
17. Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., k claffy: AS relationships, customer cones, and validation. In: IMC. pp. 243–256 (Oct 2013)
18. Marder, A., Smith, J.M.: MAP-IT: Multipass accurate passive inferences from traceroute. In: IMC (Nov 2016)
19. Prince, M.: Technical details behind a 400Gbps NTP amplification DDoS attack (Feb 2014), `http://blog.cloudflare.com/`
20. Vixie, P.: Rate-limiting state: The edge of the Internet is an unruly place. ACM Queue 12(2), 1–5 (Feb 2014)
21. Xia, J., Gao, L., Fei, T.: A measurement study of persistent forwarding loops on the Internet. Computer Networks 51(17), 4780–4796 (Dec 2007)