Behind Closed Doors: A Network Tale of Spoofing, Intrusion, and False DNS Security

Casey Deccio
Brigham Young University
Provo, UT
casey@byu.edu

Alden Hilton
Brigham Young University
Provo, UT
aldenhilton@byu.edu

Michael Briggs
Brigham Young University
Provo, UT
briggs25@byu.edu

Trevin Avery
Brigham Young University
Provo, UT
trevinavery@byu.edu

Robert Richardson
Brigham Young University
Provo, UT
richardson@stat.byu.edu

ABSTRACT

Networks not employing destination-side source address validation (DSAV) expose themselves to a class of pernicious attacks which could be easily prevented by filtering inbound traffic purporting to originate from within the network. In this work, we survey the pervasiveness of networks vulnerable to infiltration using spoofed addresses internal to the network. We issue recursive Domain Name System (DNS) queries to a large set of known DNS servers worldwide, using various spoofed-source addresses. We classify roughly half of the 62,000 networks (autonomous systems) we tested as vulnerable to infiltration due to lack of DSAV. As an illustration of the dangers these networks expose themselves to, we demonstrate the ability to fingerprint the operating systems of internal DNS servers. Additionally, we identify nearly 4,000 DNS server instances vulnerable to cache poisoning attacks due to insufficient—and often non-existent—source port randomization, a vulnerability widely publicized 12 years ago.

CCS CONCEPTS

• **Networks** → **Firewalls**; **Security protocols**; **Naming and addressing**; *Network layer protocols*; **Network measurement**.

ACM Reference Format:
Casey Deccio, Alden Hilton, Michael Briggs, Trevin Avery, and Robert Richardson. 2020. Behind Closed Doors: A Network Tale of Spoofing, Intrusion, and False DNS Security. In *ACM Internet Measurement Conference (IMC '20), October 27–29, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3419394.3423649

1 INTRODUCTION

Network administrators often use network protections such as firewalls and access control lists (ACLs) to disallow traffic from untrusted third parties from reaching internal hosts. However, source address *spoofing* creates a scenario in which inbound traffic might *appear* to be from a trusted party—even from another internal system. If traffic arriving at a given system has a source address that originates *from* that system, the legitimacy of the traffic should be questioned. This is loosely analogous to a postal service delivering a letter to an address, with the letter claiming to be *from* that address. Yet the source address of packets is often *not* checked—allowing a spoofed-source packet to penetrate a network border and reach systems not intended for public access. While the effects of this penetration can be mitigated in some cases with protocols that include some form of identity check (e.g., TCP), in other cases, this infiltration creates a vulnerability that can be exploited for surveillance or compromise.

There are two significant locations in the path of a spoofed-source packet: 1) the border of the network from which it *originates*; and 2) the border of the network for which it is *destined*. Network Ingress Filtering [20] [1]—also known as Source Address Validation (SAV) [2]—is the de facto solution for combating source address spoofing at packet *origin*, codified as Best Current Practice (BCP) 38 [20]. When spoofed-source packets are dropped as they attempt to leave their Internet Service Provider (ISP), they never become a presence in the Internet at large. However, once a spoofed-source packet reaches its *destination*, determining its validity is much more difficult—that is, unless the packet has a source IP address claiming to have originated from *within* the target network. Just as an ISP can block outbound packets that claim to have originated from outside, it can block inbound packets that claim to have originated from inside. We refer to these actions, more specifically, as *origin-side SAV (OSAV)* and *destination-side SAV (DSAV)*, respectively.

When DSAV is absent, a network is vulnerable to *infiltration*—masquerading as a network insider to penetrate a network border and access internal resources. The first major contribution of this paper is a **large-scale study of the lack of DSAV**. In late 2019, we surveyed 62,000 networks for DSAV, using methodology that was effective in its detection, yet harmless. We sent spoofed-source packets to these networks, each packet having a source appearing to originate from the network for which it was destined. We observed

[1]Note that the term "ingress" is used not because the filtering happens as a packet enters a *network* but because the filtering happens at the ingress (input) link of the participating *router*.

that about *half* of the networks we surveyed lacked DSAV, allowing our spoofed-source packets into their network.

Even more important than the fact that a network can be infiltrated is the *impact* of the unauthorized access—how it might be *exploited* to survey or compromise internal systems. The second major contribution of this paper is an **analysis of internal systems** reached via network infiltration, as a case study motivating the importance of DSAV. We characterize and assess the vulnerability of systems accessed through experimental spoofed-source packets. We accomplished this by issuing spoofed-source Domain Name System (DNS) queries to almost 12 million DNS servers. These queries reached about 5% of targets, allowing us to survey over a half million servers. Within the target networks lacking DSAV, we identified nearly 4,000 DNS servers that were vulnerable to cache poisoning attack, 59% of which would have been protected had DSAV been in place. While untested as part of this work, networks lacking DSAV also expose otherwise unreachable—and possibly vulnerable—DNS resolvers to other attacks such as DNS zone poisoning [29] and the recently disclosed NXNS attack [43].

2 BACKGROUND AND PREVIOUS WORK

BCP 38 [20] urges ISPs to deploy OSAV to prevent packets with spoofed sources from leaving their networks. This containment prevents these networks from being contributors to spoofing-based attacks, such as *reflection* and *intrusion*. In a reflection attack, an attacker spoofs the address of a victim in the request to a server, and the server sends its response to the victim, typically in another network. Network intrusion occurs when a network with no OSAV sends a spoofed-source packet to a network with no DSAV, and the packet's source matches IP prefixes originated by its target network. In this case, packets enter a network with a spoofed source that appears to have originated from within the destination network. The internal system receiving the spoofed-source packets will see them as having originated internally.

Spoofing-based reflection and intrusion must be carried out with an application-layer protocol, such as the DNS. The DNS is a query-response protocol used for translating domain names to Internet resources, such as IP addresses. *Stub resolvers* query *recursive resolvers* (or servers), which find an answer by querying *authoritative servers*. While authoritative DNS servers are typically open to queries from any Internet entity, recursive servers are traditionally *closed*, only allowing queries from known clients [10]. *Open* recursive servers exist, but are discouraged (although public DNS services are becoming more prevalent [8, 22, 23, 39, 45]). For this reason, authoritative DNS services and open recursive DNS services are more likely to be used in reflection, whereas closed resolvers are the more likely target for intrusion.

In the area of DNS-based reflection attacks, researchers have explored both attack potential [44] and the deployment of DNS Response Rate Limiting (RRL) [12, 33, 34, 46].

With regard to SAV measurement, the Spoofer Project has been involved in measuring SAV for over 10 years [2, 5, 32]. Spoofer data comes from participants who voluntarily install and run the Spoofer client on their machines. This client's primary role is to send out spoofed-source probes to test for OSAV. However, it also plays a part in DSAV measurement. Spoofed-source packets appearing to originate from the client's network are sent to each client. If the client receives the spoofed packet, Spoofer is able to infer the lack of DSAV. Researchers recently reported a surprising 67% and 74% of the IPv4 and IPv6 autonomous systems (ASes) measured, respectively, lacking DSAV [32].

Our approach addresses two limitations associated with Spoofer's DSAV measurement. First, a significant portion of the Spoofer clients are run behind Network Address Translation (NAT) gateways, for which DSAV cannot be tested (i.e., because the client has no public IP address from which it can be reached). Secondly, Spoofer requires a user to opt in to the study by downloading and running the client. Our methodology only targets public IP addresses on existing infrastructure, such that no client software is required.

In work performed concurrently with (and independently of) our own, Korczyński, et al. [30], tested networks for source address validation using a methodology similar to ours. They issued queries to every IP address in the IPv4 space, in each case spoofing the source IP address just higher than the selected destination. In contrast, our objectives focused on exploring the variety of ways in which a lack of DSAV might be discovered. Our methodology differs primarily in that 1) we use as many as 101 diverse, spoofed-source IP addresses for each destination, rather than only the next sequential IP address; 2) the selection of target IP addresses used in our research consist of those that generate query activity at the root servers; and 3) our study includes both IPv4 and IPv6. Our results show that there are advantages to both the current methodology and that used by Korczyński, et al. In particular, the sheer breadth of the IPv4 address space scanned by Korczyński, et al., resulted in more overall hits than our targeted approach. The diversity of spoofed sources used in our experiment uncovered resolvers—and ASes—that would not have otherwise been identified using only a same-prefix source. We discuss this further in Section 4.1. Nonetheless, the overall percentage of measured ASes with reachable IPv4 targets is consistent between the two studies, within 1%: 48.78% vs. 49.34%. Finally, in the current paper, we extend our analysis to survey and identify vulnerabilities of internal systems, as a case study of our methodology.

Related to our vulnerability analysis of internal DNS resolvers, Scheffler, et al. [42], analyzed internal DNS recursive servers using a different technique. By communicating over the Simple Mail Transfer Protocol (SMTP) with servers that performed Sender Policy Framework (SPF) [28] validation, they were able to elicit queries of the mail servers' recursive DNS servers. Their results turned up very little evidence of servers lacking source port randomization, whereas our study shows a non-trivial number—nearly 4,000 DNS resolvers with invariant source ports.

3 EXPERIMENT SETUP

Our DSAV experiment consisted of sending DNS queries with spoofed-source IP addresses to millions of recursive DNS servers world-wide. Our goal was to determine whether or not each query reached its target DNS server, and thus penetrated the network border in the process. Having no observable presence at the addresses we spoofed, we had no way of assessing reachability by examining responses. Thus, we determined that a recursive DNS server was

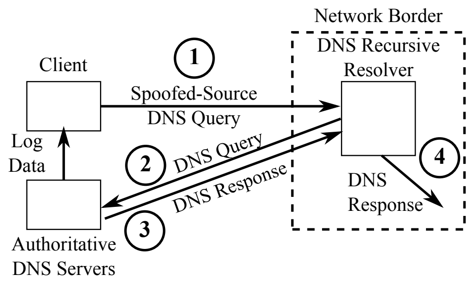


Figure 1: Experiment setup, in which (1) a client sends a DNS query with spoofed source to an internal DNS recursive resolver, (2) the recursive resolver issues a query to our DNS authoritative servers, (3) the authoritative server responds, and (4) a DNS response is issued by the DNS recursive server.

reachable if, for a given query, we observed a corresponding query at a DNS authoritative server—an indicator of the recursive server's attempt to resolve the query name. The query names used in our experiment were such that 1) no query name would ever be found in the cache of a DNS resolver and 2) the servers authoritative for all queries are under our control (see Section 3.3). Figure 1 illustrates our setup.

3.1 DNS Servers (Targets)

To generate a set of target IP addresses, we used the "Day in the Life" (DITL) [16] data, sponsored by the DNS Operations, Analysis, and Research Center (OARC) [17]. The DITL data consists of 48 hours of DNS queries destined for the DNS root servers, contributed by members of the DNS operator community, including operators of the DNS root servers [41]. Thus, it provides a rich source of potential recursive DNS servers for our experiment. We extracted the source IP addresses from the DNS queries captured in the 2019 DITL collection (April 2019).

Not all of the source IP addresses extracted from the DITL data were acceptable targets for our experiment. We excluded about 4 million addresses designated as "special purpose" addresses by IANA [9]; for these addresses, there would be no legitimate entries in public routing table. We excluded another 36,027 source IP addresses for which there was no announced route from which we could derive other-prefix addresses from the same AS (see Section 3.2). Ultimately, our set of target IP addresses consisted of 11,204,889 IPv4 addresses and 784,777 IPv6 addresses, from 53,922 and 7,904 ASes, respectively.

3.2 Spoofed Sources

The set of source addresses for each target was selected with the intent of maximizing the chances that the target would accept a query. If the server rejected a query, we could not systematically know if that query had penetrated a network border. For any given target IP address, we issued as many as 101 DNS queries using spoofed sources from the following categories:

- *Other prefix*: up to 97 other-prefix addresses (explained hereafter).
- *Same prefix*: an IP address from the *same* /24 (IPv4) or /64 (IPv6) prefix.
- *Private or unique local*: 192.168.0.10 or fc00::10.
- *Destination-as-source*: the target IP address itself.
- *Loopback*: 127.0.0.1 or ::1.

The other-prefix addresses were generated as follows. We first looked up the AS number (ASN) for every target IP address. For each ASN in the resulting set, we looked up all the IP prefixes originating from that AS. The next steps depended on whether the IP address and ASN were associated with IPv4 or IPv6.

For IPv4, we divided all the IP address space originating from an AS into 24-bit prefixes. Every /24 prefix containing a target IP address was set aside for random selection of a same-prefix IP address. From each of the remaining /24 prefixes, we selected, at random, a single IP address. In both cases, the first and the last IP addresses were excluded from selection because of their reserved status in a /24 subnet. The resulting IP addresses formed the set of other-prefix addresses for any target IP address announced by that AS. Because some ASes had a prohibitively large number of /24 prefixes, we limited our selection to 97 prefixes[2].

For IPv6, we similarly divided each AS's aggregate IP address space, but we used a 64-bit prefix, which is the typical prefix length for IPv6 subnets. As with IPv4, we selected an IP address from within each /64 prefix announced by the AS—for both the same-prefix and other-prefix addresses. However, we used a more targeted methodology to identify more realistic client addresses, rather than blindly probing the sparsely-populated IPv6 address space. First, for IPv6 prefix selection, we gave preference to /64 prefixes that contained IPv6 addresses from an IPv6 "hit list" [21]—a sign of observed activity within that prefix. Second, address selection within a /64 prefix was limited to the first 100 addresses in the /64 prefix (minus the first two addresses, often used for the router address).

3.3 Query Names

We encoded the query names used in our experiment according to the following template: **`ts.src.dst.asn.kw`**`.dns-lab.org`, where **ts** is the timestamp the query was sent, **src** is the spoofed-source IP address, **dst** is the target IP address, **asn** is the ASN of the target IP address, and **kw** is a keyword associated with the current experiment. With this template, we could associate any query arriving at the `dns-lab.org` authoritative servers (under our control) with the experimental query that induced it. The use of a timestamp in the query name ensured the uniqueness of a given query such that it would never be in the cache of a recursive resolver.

Our primary interest was whether a query with one of the experimental query names reached our authoritative DNS servers. We did not see any particular benefit to making the experimental query names actually resolve. Therefore, to simplify our setup, our authoritative servers returned an `NXDOMAIN` (name does not exist) response code in response to any experimental queries. When analyzing our data, however, we learned that there were some side effects associated with this approach that caused some gaps in our

[2]The number 97 was chosen when we had three other spoofed-source categories in mind, such that the maximum number of source IP addresses we would use for a given target would be an even 100. However, we ended up adding another source IP address to our experiment, such that at most 101 sources would be used to query a given target.

experiment visibility. We discuss these side effects and quantify their impact on our analysis in Section 3.6.4.

3.4 Query Execution

The queries were scheduled such that the entire experiment would be carried out in about four weeks time. Considering the number of target IP addresses and the sources for each, the rate of DNS queries leaving our client maintained a rate of roughly 700 queries per second—which was an administrative constraint we were required to work with. We spaced the queries for a given target IP address such that they were evenly spread over the entire duration of the experiment.

The queries associated with our experiment were issued between November 6 and December 27, 2019, from a network that lacked OSAV (BCP 38 [20]). The absence of OSAV in our client's network was a requirement for effectively testing DSAV. The time frame for our experiment was longer than the four weeks we had planned because of several unexpected interruptions, including a power outage. Despite the gaps in our experimental activity, we were able to successfully issue all of the prepared queries associated with the experiment, albeit behind schedule.

3.5 Follow-Up Queries

We monitored authoritative DNS server logs to detect incoming queries that had been generated as a result of our activity, in real time. Whenever we *first* observed a DNS query associated with a given target IP address, a series of follow-up queries were issued to that target IP address. The follow-up queries were sent using the same spoofed-source address as that which induced the query first observed at our authoritative servers. Subsequent queries associated with the target IP address (i.e., beyond the first) were logged but not further acted on; thus, a given target IP address only received one set of follow-up queries. The follow-up queries included:

- *IPv4- and IPv6-only*: two sets of 10 queries that elicited queries exclusively over IPv4 and IPv6, respectively, to our authoritative servers.
- *Open resolver*: non-spoofed-source query.
- *TCP*: a query that elicited a DNS-over-TCP query to our authoritative servers.

The IPv4- and IPv6-only queries were elicited by using query names in DNS domains that were only delegated to IPv4 addresses or IPv6 addresses, respectively. The TCP query was elicited by issuing a query for which the authoritative server would always respond with the truncation (TC) bit set. A truncated response causes the recursive resolver to issue its query again over TCP [13].

3.6 Methodology Considerations

Several issues merit our discussion, including the effect of middleboxes or human intervention on our experiment, data freshness, and QNAME minimization.

3.6.1 Middleboxes. It is possible for a DNS request to be transparently intercepted and handled by a middlebox between our client and the target DNS resolver [6]. In this case it would be unclear if the DNS resolver itself—or, more importantly, its network—was reachable. We observed that for 86% of IPv4 ASes and 95% of IPv6 ASes, at least one recursive-to-authoritative query was received directly from an address in the target AS. In these cases, even if our spoofed-source query did not reach its destination IP address, we know that reached its destination AS, which confirmed lack of DSAV. As for the remaining ASes, at least one recursive-to-authoritative query was received from major public DNS services (Cloudflare [23], Google [22], CenturyLink [7], OpenDNS [8], or Quad9 [39]) for 89% of IPv4 ASes and 86% of IPv6 ASes. Forwarding to such DNS services is not characteristic of middleboxes. These numbers explain all but 2% (IPv4) and 1% (IPv6) for our per-AS DSAV measurements.

3.6.2 Data Freshness. We consider the reasonableness of using DITL as a set of target recursive IP addresses. Certainly not all of the approximately 12 million target IP addresses from the DITL data were functioning as recursive servers at the time of our experiment. It is possible that some IP addresses that *did* represent recursive DNS servers at the time of DITL collection, *no longer did* when we ran our experiment six months later. Previous research has shown that there is churn in IP addresses of DNS resolvers—specifically, open resolvers—over time [31]. It is also possible that some of the IP addresses might never have been used used as recursive DNS servers. For example, perhaps they represent software used only to monitor Internet connectivity or health. We argue that we are working with an incomplete but sufficiently representative data set. This is validated in part by the fact our results are consistent with those from previous work (see Section 2). Additionally, we do not expect our data to include the IP addresses of *all* recursive servers. This is in part because the DITL data is not comprehensive (i.e., not all root servers participate in the collection) and in part because an active resolver might not need to query the root server during that period, depending on its query patterns and caching behaviors. Finally, the source IP address of some of the queries captured in the DITL data might in fact be spoofed and thus not associated with an actual DNS resolver.

There is also the question of whether or not IP churn occurring during the experiment itself affected the results, causing addresses that were at one point responsive to be unresponsive later on. While this situation is certainly possible, it could really only affect the results of one aspect of our study, that of spoofed-source effectiveness (Section 4.1); for the rest of our evaluation of DSAV, an AS was considered to be lacking DSAV if at least *one* query was handled by a target IP address. We might have chosen to send all queries to a given target IP address in rapid succession, rather than spreading them out over the entire experiment period; this would have mitigated the question of churn. However, we opted to minimize possible impact and attention, e.g., from IDS, by spreading the queries out.

3.6.3 Human Intervention. We expect that some of our spoofed-source queries might be dropped and/or logged by IDS or servers as suspicious. Curious human analysts might resolve the domain name to learn more about the activity, resulting in a query to our authoritative servers. However, such a query does not provide reliable DSAV information.

To overcome this ambiguity, we calculated query *lifetime* (i.e., how long it was "alive" in the system) by subtracting the timestamp embedded in the query name from the time at which the query was received at our authoritative servers. We considered a query with a lifetime of 10 seconds or less as unlikely made by a human, in response to logs. While a query passing through the most reliable systems might consistently have a lifetime of less than a single second, we selected this higher threshold because query retransmissions (i.e., by recursive resolvers) can happen after timeouts of one or more seconds. The results we present only include queries whose lifetime was under our 10-second threshold. Queries for an additional 3,444 IPv4 addresses and 70 IPv6 addresses had a lifetime that exceeded our threshold, representing less than 0.1% of addresses, for both protocols. These corresponded to 421 IPv4 ASes and 32 IPv6 ASes. For all but 19 and 2 of these ASes, we were able to infer lack of DSAV through the presence of other resolvers which did query our servers within the 10 second window.

3.6.4 QNAME Minimization. In an effort to preserve privacy, some modern DNS resolvers avoid sending authoritative servers the full query name (QNAME) and instead only ask for the next unknown label. This is known as *QNAME Minimization* [3]. In the case of our experiment, before asking for the full QNAME (i.e., `**ts.src.dst.asn.kw**.dns-lab.org`), a resolver using QNAME minimization would ask for `**kw**.dns-lab.org`, then `**asn.kw**.dns-lab.org`, etc. As we mentioned in Section 3.3, our authoritative servers returned an `NXDOMAIN` response code in response to any queries related to our experiment. For at least some resolver implementations that implement QNAME minimization, an `NXDOMAIN` response halts further queries associated with the QNAME. This is because an `NXDOMAIN` for a given domain name implies that no subdomains (i.e., with additional labels on the left) exist [4].

We observed QNAME-minimized queries from 17,981 (0.16%) of the IP addresses that we targeted with our initial reachability query. For 9,898 (55%) of these IP addresses, we never received a query with the full QNAME. Most notably, they did not include the label with the encoded source address. With no way to identify the source IP address that we used to reach these 9,898 targets, we exluded them from the total number of reachable targets. Nonetheless, we still learned something about the networks from which the QNAME-minimized queries originated. We observed that DNS clients from 2,081 ASNs queried for `kw.dns-lab.org` (the product of QNAME minimization). Of those, 2,041 (98%) were identified as lacking DSAV, in that we observed queries from these same QNAME-minimizing resolvers or from other (i.e., non-QNAME-minimizing) resolvers. Thus, QNAME minimization did not diminish our DSAV measurement results. Nonetheless, a future version of our experiment would produce more inclusive results by returning answers synthesized from wildcard entries, rather than returning `NXDOMAIN`.

3.7 Ethical Considerations

The measurement of network and systems vulnerabilities requires care, both in the activity itself and in the disclosure of the findings. Because of the nature of our research, we consulted various resources for ethical guidance. While our organization has no ethics board, we consulted with individuals from the legal department, the office of research and creative activities, and our own computer science department with respect to the ethics of our research. We likewise reviewed the Menlo Report [14], which holds some of the key guidelines for ethical research in this area. Of the ethics principles outlined in the Menlo Report, those most applicable to our current research are 1) justice, 2) respect for law and public interest, and 3) beneficence.

Regarding justice, our measurements considered all target IP addresses (i.e., from the DITL data) equally; no particular industry, geography, nation state, address space, or protocol was deliberately targeted more than another.

Perhaps the biggest ethical question associated with our research was the legality of measuring another's network using our methodology. Our measurements crossed interstate boundaries world-wide, each potentially with their own laws regarding unauthorized network access. For example, the United States (U.S.) outlaws any intentional access of non-public, government-owned computer systems, without authorization [24]. We cannot definitively determine whether or not the systems that we measured are non-public nor whether our benign packets even constitute a violation of this statute. In any case, we believe that our methodology is justified because of the benefit it brings in the public interest. Indeed the Menlo Report's principle of beneficence suggests that the benefits of an experiment should be maximized and the harms minimized. Bringing to light the severity and pervasiveness of the lack of DSAV and the potential for network penetration is extremely valuable to the Internet community. We expect that responsibly publishing our findings will be a catalyst in spreading awareness and taking the necessary action to fill the security gaps identified herein.

The potential harms associated with our experiment might include degradation of service due to our traffic, time spent following up on alerts from Intrusion Detection Systems (IDS), or careless vulnerability disclosure. We took several measures to minimize any negative impact, and even the appearance of abuse. First, we limited both the number and the rate of queries directed towards any given destination, as described in Section 3.4. Considering query rates at production DNS servers are typically measured in queries per second, and our maximum per-destination rate was on the order of four per day—plus a one-time series of fewer than 30 follow-up queries—the impact of our experiment would have been barely, if at all, noticeable. Second, the `SOA` (start of authority) record of the DNS zone corresponding to our query names (see Section 3.3) included: 1) a `RNAME` (responsible name) field with an email address with which we could be contacted, e.g., for more information or to opt out; and 2) an `MNAME` (master server name) field with the domain name of a Web sever providing a brief description of this project. The project description included contact and opt-out information. The system from which the queries with non-spoofed sources were sent (see Section 3.5) also ran a Web server with the same project information.

3.8 Response and Opt Out

Despite the approximately 1 billion queries that we sent to nearly 12 million target IP addresses, we received just five communications related to our experiment. All respondents requested to be opted out of our experiment. In each case, we obliged their request and stopped sending spoofed queries to their address space. When they requested, we also offered more information about our experiment and some of our initial findings. After learning more about our work,

	ASes		IP targets	
Country	**Total**	**Reachable**	**Total**	**Reachable**
United States	16,782	4,675 (28%)	2,926,342	93,993 (3.2%)
Brazil	6,468	3,803 (59%)	396,978	19,156 (4.8%)
Russia	4,937	2,917 (59%)	361,763	42,026 (11.6%)
Germany	2,470	887 (36%)	997,994	38,190 (3.8%)
United Kingdom	2,246	745 (33%)	405,850	18,360 (4.5%)
Poland	2,041	1,064 (52%)	119,275	7,136 (6.0%)
Ukraine	1,709	1,076 (63%)	68,427	10,545 (15.4%)
India	1,592	649 (41%)	336,834	38,983 (11.6%)
Australia	1,562	507 (32%)	177,717	8,233 (4.6%)
Canada	1,519	553 (36%)	297,534	8,397 (2.8%)

Table 1: DSAV results for the 10 countries associated with the largest fraction of ASes (IPv4 and IPv6) in our set of target IP addresses.

	ASes		IP targets	
Country	**Total**	**Reachable**	**Total**	**Reachable**
Algeria	15	6 (40%)	15,867	11,627 (73%)
Morocco	22	10 (45%)	24,895	13,189 (53%)
Eswatini	7	6 (86%)	636	281 (44%)
Belize	30	12 (40%)	1,332	555 (42%)
Burkina Faso	14	6 (43%)	1,280	498 (39%)
Kosovo	5	3 (60%)	49	18 (36.7%)
Bosnia & Herzegovina	48	26 (54%)	5,008	1,524 (30%)
Seychelles	25	11 (44%)	793	241 (30%)
Wallis & Futuna	1	1 (100%)	11	3 (27%)
Ivory Coast	15	8 (53%)	6,609	1,731 (26%)

Table 2: DSAV results for the 10 countries associated with the largest percentage of IP addresses (IPv4 and IPv6) reachable by spoofed-source packets.

one inquiring administrator actually wrote back and requested that his address space be put back into our set of targets!

From four of the five communications we received, it was evident that our spoofed-source query reached its target DNS resolver, but that the DNS resolver refused to handle the query (i.e., with a `REFUSED` response code). This was evidence that our findings are in some ways a conservative estimate of the pervasiveness of the lack of DSAV. Even with a pool of spoofed sources to select from, we were unable to find a legitimate source for every server that we reached.

4 DSAV EXPERIMENT RESULTS

The pervasive lack of DSAV well exceeded our expectations. Of the 11,204,889 IPv4 addresses targeted with our experiment, at least 519,447 (4.6%) received and handled one or more of our queries, as indicated by a recursive-to-authoritative query observed at our authoritative DNS servers. Similarly, of the 784,777 IPv6 addresses we targeted, 49,008 (6.2%) recursively handled at least one of our spoofed-source queries. While the figures for target IP addresses are nominal, the number of ASes with affected IP addresses was far more pervasive: 26,206 (49%) of 53,922 IPv4 ASes and 3,952 (50%) of 7,904 IPv6 ASes were vulnerable to infiltration via spoofed-source packets. These numbers represent the lower bound of networks that do not support DSAV.

Table 1 and Table 2 list the 10 countries representing the *most ASes* in our data set and those having the *highest percentages of IP addresses accepting spoofed-source packets*, respectively. The country for each IP address was looked up using MaxMind's GeoLite2 data [35], and each AS was associated with one or more countries based on the GeoIP data for its constituent IP addresses. Thus, an AS might be counted multiple times in different countries. The data set included over twice as many United States (US)-based ASes than the next most represented country, Brazil. Nevertheless, the diversity of percentage of ASNs lacking DSAV (i.e., "reachable") is apparent, with the US being below average at 28%, and Brazil, Russia, and Ukraine showing over that over half of ASes lack DSAV. In terms of reachable target IP addresses, Algeria and Morocco topped the list, each with over 50% of targeted IP addresses having received our DNS queries.

4.1 Spoofed Source Effectiveness

We now consider the effectiveness of the (up to) 101 sources that were used, in terms of eliciting DNS activity of their target. All numbers are presented as a fraction of the total *reachable* targets.

We first analyze the overall fraction of spoofed sources that reached their targets. For nearly half of all reachable target IP addresses (collectively IPv4 and IPv6), only one or two sources resulted in reachable queries. The median number of spoofed sources with which queries reached IPv4 and IPv6 destinations was 3 and 2, respectively. However, 16% of IPv4 destinations and 9% of IPv6 destinations were reachable using over 50 spoofed sources.

Next we analyze the effectiveness of the different categories of spoofed sources (see Section 3.2), in terms of DSAV detection. Spoofed sources from every category reached at least one IP target. The breakdown of target IP addresses and ASNs reachable by spoofed source category is shown in Table 3, under the heading "Category-Inclusive." For IPv4, other-prefix and same-prefix sources were the most prevalent, reaching 78% and 63% of all IPv4 targets that received spoofed-source queries. The other-prefix category was also well represented for IPv6 targets, reaching 45% of all reachable IPv6 targets. However, the IPv6 sources that dominated reachability were in the same-prefix and destination-as-source categories, reaching 84% and 70% of reachable IPv6 targets, respectively. It was notable that the percentage of target-reaching queries that used destination-as-source was much higher for IPv6 targets (70%) than that for IPv4 hosts (17%). We observed in our lab testing that modern Linux kernels drop destination-as-source packets that use IPv4, but IPv6 destination-as-source packets are sent to user space (see Section 5.5). Thus, if the fraction of reachable Linux targets is equal between the IPv6 and IPv4 realms, we would expect the percentage of IPv6 destination-as-source hits to be higher. Private-address sources also had a presence, reaching 3.4% of reachable IPv4 targets and 4.3% of reachable IPv6 targets, respectively.

	Category-Inclusive (one or more)				Category-Exclusive (only)			
Source	IPv4 Targets		IPv6 Targets		IPv4 Targets		IPv6 Targets	
Category	Addresses	ASNs	Addresses	ASNs	Addresses	ASNs	Addresses	ASNs
All Queried	11,204,889	53,922	784,777	7,904	11,204,889	53,922	784,777	7,904
All Reachable	519,447 (4.6%)	26,206 (49%)	49,008 (6.2%)	3,952 (50%)	519,447 (4.6%)	26,206 (49%)	49,008 (6.2%)	3,952 (50%)
Other Prefix	405,018 (78%)	25,376 (97%)	22,073 (45%)	3,388 (86%)	172,372 (33%)	1,808 (6.9%)	2,410 (4.9%)	152 (3.9%)
Same Prefix	327,000 (63%)	23,895 (91%)	40,986 (84%)	3,556 (90%)	90,366 (17%)	335 (1.3%)	3,972 (8.1%)	55 (1.4%)
Private	17,762 (3.4%)	3,078 (12%)	2,098 (4.3%)	544 (14%)	2,508 (0.5%)	110 (0.4%)	229 (0.5%)	19 (0.5%)
Dst-as-Src	89,281 (17%)	12,400 (47%)	34,311 (70%)	3,179 (80%)	13,384 (2.6%)	202 (0.8%)	4,869 (9.9%)	182 (4.6%)
Loopback	1 (0.0%)	1 (0.0%)	106 (0.2%)	26 (0.7%)	0 (0.0%)	0 (0.0%)	22 (0.0%)	4 (0.1%)

Table 3: Number of IP addresses or ASNs for which at least one spoofed-source reached its target ("Category-Inclusive") or for which a spoofed-source category was the *only* one to reached its target ("Category-Exclusive"). Percentages in the "All Reachable" row represent the fraction of *all* targets queried, whereas other percentages in other rows represent the fraction of *reachable* targets.

Each category of spoofed source *independently* contributed to the overall effectiveness of our experiment. *Every* category resulted in reaching a target that would not otherwise have been reached, even considering all other categories combined. Thus, if we had excluded any category of spoofed addresses from our experiment, our total number of reachable targets would have been lower—both by IP address and ASN. This is shown in Table 3, under the heading "Category-Exclusive." Notably, had we limited our spoofed sources to addresses within the same IPv4 /24 (or IPv6 /64) as the target, with a source address distinct from the destination address, we would not have discovered 37% of reachable IPv4 addresses or 9% of reachable IPv4 ASNs. While the query with spoofed loopback source was handled by relatively few targets, 22 IPv6 addresses and four ASNs would not have resulted in hits without our inclusion of this source category.

5 DSAV CASE STUDY: DNS RESOLVERS

While the knowledge that a network lacks DSAV is valuable in and of itself, in this section we demonstrate how that knowledge might be used by someone with malicious intent to survey or exploit vulnerabilities of internal systems.

5.1 Closed Resolvers

Per RFC 5358, "by default, nameservers SHOULD NOT offer recursive service to external networks" [10]. The two primary reasons for this are safety for others and safety for self. Open resolvers can be used as unwitting accomplices in attacking *others* in reflection-based distributed denial-of-service (DDoS) attacks. Additionally, open resolvers *themselves* are more exposed to various attacks which are facilitated by the attacker's ability to induce recursive-to-authoritative queries, such as DNS cache poisoning [27]. Thus, one of the IANA's first actions in the wake of the Kaminsky attack disclosure was to reiterate the need to disable open resolvers [11].

Using the open resolver query (Section 3.5), we classified the resolvers we reached as *open* if we observed a recursive-to-authoritative query in response to the non-spoofed-source query, and *closed* otherwise. In total, we classified 340,247 (60%) resolvers as closed and 228,208 (40%) as open. One unfortunate phenomenon confirmed by these results is that a large number of resolvers continue to operate open to the public, which mirrors the results of recent studies [31, 38]. However, we emphasize that even reaching an open resolver with our experiment indicates a lack of DSAV—i.e., because our spoofed-source query arrived at its destination. The fact that the resolver responds to *any* source address is simply an additional layer of insecurity.

That said, closed resolvers are of particular interest. Because an attacker can induce recursive-to-authoritative queries with spoofed-source queries, closed resolvers in a network that lacks DSAV have little advantage over open resolvers when it comes to cache poisoning attacks. The big difference is that the closed resolvers are *thought* to be limiting their query access to trusted clients. The closed resolvers we identified are all stripped of their first line of defense against cache poisoning attacks, all due to their networks' failure to employ DSAV. Additionally, despite the large presence of open resolvers, at least one *closed* resolver was reached in 88% of ASes that we identified as lacking DSAV. That is, nearly 9 out of 10 networks host a DNS resolver that is thought, *falsely*, to be limiting its query access.

5.2 Source Port Randomization

One of the biggest questions driving the current research was whether systems that were thought to be inaccessible due to firewall or access control list exhibited some vulnerability that is less likely to be seen in a publicly accessible system. To this end, we investigated one of the most prominent DNS resolver vulnerabilities in recent years: lack of source port randomization [18, 27]. While this vulnerability was disclosed and related patches were distributed nearly 12 years prior to the writing of this paper, we hypothesized that there might be some instance of it in the wild, behind closed doors.

For this part of our analysis, we only analyzed behavior of DNS resolvers that contacted our authoritative servers directly, i.e., the source IP address of the query matched the **dst** label of the query name. This allowed us to analyze the systems associated with the target IP addresses, not the systems they might forward to (see Section 5.4).

To assess the source port randomization of the reachable targets, we computed the range of source ports for the 10 IPv4 or IPv6 follow-up queries (see Section 3.5) observed for each target IP address. The

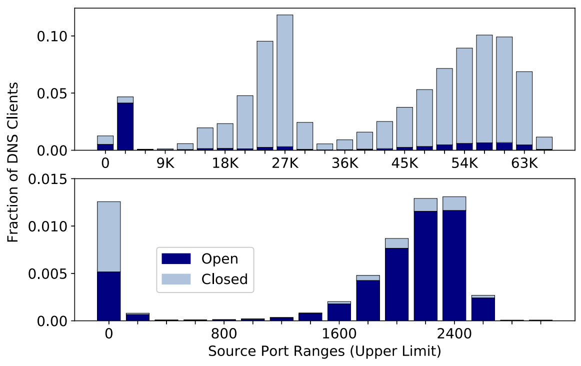


Figure 2: Frequency distribution of source port ranges used by resolvers reachable with spoofed-source query. The upper plot shows the frequency of ranges between 0 and 65,535 (max), and the lower plot shows the frequency of ranges 0 through 3,000 (i.e., a zoomed-in version of the upper plot). Each bar is broken down by open-closed resolver status.

source port range is a useful heuristic to identify resolvers that are not randomizing their source ports. It also provides characteristics that are helpful for identifying OS or software (see Section 5.3).

Figure 2 shows two histograms representing the frequency distribution of source port ranges for the DNS resolvers. The upper plot shows the frequency of ranges between 0 and 65,535 (max), and the lower plot shows a zoomed-in version of the upper plot, covering ranges 0 through 3,000. Additionally, each bar is broken down by its makeup of open and closed resolvers. There are several prominent characteristics in Figure 2 that we discuss in the following sections.

5.2.1 Zero Source Port Randomization. We first focus our attention on the DNS resolvers with a source port range of zero, i.e., the left-most bar in either plot of Figure 2. In response to our spoofed-source follow-up queries, 3,810 resolvers issued 10 queries to our authoritative servers with *no* variance in source port! These account for 1,802 (6%) of all ASes (IPv4 and IPv6) lacking DSAV. It would be trivial to poison the cache of these resolvers due to the combination of 1) the capability to induce a query through spoofed-source query and 2) no source port randomization. With a known source port, only the query's 16-bit transaction ID is left to guess; the search space is reduced from 2^{32} (4.3 billion) to 2^{16} (65,536). More than half of these resolvers (2,244 or 59%) are closed, meaning that DSAV would reduce their attack potential.

Lack of source port randomization is rooted in two primary causes: old software and improper configuration. While most modern DNS software implementations use random source ports for DNS queries, many older versions used only a single source port. For example, the Berkeley Internet Name Domain (BIND) version 8 used a unprivileged port (> 1023), as did versions of Windows DNS prior to 2008 R2. Versions of BIND prior to BIND 8.1 used port 53 exclusively [26]. Even when default software behaviors changed such that random source ports were used, custom configuration options allowed the updated software to designate a single source port. This option was even included in the configuration file distributed with the BIND package on at least one OS. However, it was removed in 2008, in conjunction with the source port vulnerability disclosure [18, 26].

Due to its historical roots in both software defaults and configuration, port 53 was observed more than any other source port, being used by 1,308 (34%) of the single-source-port resolvers that we reached. Other frequently-used source ports were 32768 (12%) and 32769 (3.8%).

To better understand the cause for the lack of source port randomization, we contacted administrators of several resolvers that exhibited this behavior. To find administrator contact information, we performed a reverse DNS (PTR) lookup of the IP address for each resolver and then looked up the SOA record for the domain of the DNS name returned. We used the RNAME (responsible name) field of the SOA record as a contact email address for the affected resolver(s). We then selected 43 administrators, representing 53 (1.4%) of the resolvers that exhibited fixed source port behavior: 40 were selected randomly—half associated with resolvers using source port 53 and half associated with resolvers using an unprivileged source port—and 3 were administrators with whom we had prior acquaintance. We received responses from just five administrators, all associated with systems that used source port 53 exclusively. Three of the respondents were the administrators that we knew. Two of the five confirmed a BIND configuration using fixed source port; the others provided no comment on their configuration. However, three of the five administrators communicated to us that their DNS resolvers were configured to only handle queries from within a designated IP prefix, previously unaware that their systems were reachable with spoofed-source queries. Anecdotally this confirmed that less-secure configurations might be tolerated—or even defended—on systems that are thought to be unreachable by untrusted parties.

5.2.2 Passive Measurement Comparison. To increase confidence in our active measurement analysis, we compared it to passive measurements for the same resolvers. Using the 2018 DITL data [15] (the 2019 DITL data was inaccessible at the time we ran our analysis), we collected query information for all of the DNS resolvers that exhibited zero-range source port behavior. To decrease the likelihood of false positives, we considered only IP addresses for which we observed queries that would provide a fair comparison to our active measurement. Thus, an IP address was included in our analysis only if we observed one or both of the following from that IP address: 1) 10 queries for unique query names or 2) one or more queries exclusively using the same source port that we observed in the follow-up queries associated with our active measurement. We analyzed the range of the source ports used by each IP address over the 48-hour period covered by the 2018 DITL collection.

The results tell an interesting story. First, 1,954 (51%) of the 3,810 DNS resolvers that exhibited no source port variance in connection with our follow-up queries similarly showed no variance in 2018. Perhaps more alarming is that 959 (25%) of the DNS resolvers *currently* exhibiting no source port variance had at least *some* element of source port variation 18 months earlier. That is, their vulnerability has actually increased in the past couple of years. The 2018 DITL data did not include sufficient data for us to compare 897 (24%) of the resolvers that we observed using a single source port.

5.2.3 Ineffective Source Port Allocation. We now turn our attention to the non-zero, low-numbered ranges observed in our data. There were 244 DNS resolvers in 142 ASNs with a source port range between 1 and 200. For the resolvers in this category, 159 (65%) yielded source ports that followed a strictly increasing (i.e., non-random) pattern; of those, 130 wrapped after reaching some some maximum value. For 34 (14%) of the 244 resolvers, only seven or fewer unique port values were observed out of the 10 total queries, a phenomenon that would typically only occur 0.066% of the time—or 1 out of every 1,500—if the size of the pool being selected from was actually 200. The cases of both non-random port and small port pool appear to violate RFC 5452, which requires "an unpredictable source port for outgoing queries from the range of available ports. . . that is as large as possible and practicable" [19].

5.3 OS Identification

As part of our case study, we now consider how internal systems might be surveyed to identify OS and DNS software. While this knowledge might be valuable in and of itself, sometimes it could expose other vulnerabilities to an attacker. For example, the SigRed attack was recently disclosed as an attack on all versions of Windows DNS [36].

Using only the query data associated with our experimental DNS query activity, we were able to infer OS and DNS software. The primary methods we applied were: 1) the `p0f` fingerprinting tool and 2) observed source port ranges. We applied these methods both individually and in support of one another. We also used an experimental lab environment to empirically learn behaviors and validate our methods.

5.3.1 `p0f`. We used `p0f` to analyze the TCP/IP packets associated with the DNS-over-TCP queries that were elicited by our TCP follow-up queries (see Section 3.5). `p0f` uses packet characteristics, such as IP time-to-live and TCP maximum segment size, to associate packets with the operating system that produced them. Windows and Linux systems were among those identified by `p0f`, consisting of 5.4% and 2.5% of DNS resolvers, respectively. However, `p0f` was only able to categorize about 10% of our resolvers; 90% remained unclassified.

While `p0f` struggles to identify the OS of the majority of DNS targets, it tells us something of those resolvers with a source port range of zero. According to `p0f`, the TCP/IP fingerprint of 760 (20%) of these DNS resolvers matched the characteristics of BaiduSpider, the Web crawler associated with the prominent Chinese search engine, Baidu [1]. Another 451 (12%) were identified as Windows systems—likely pre-2008 R2, for which using a single port is the default. Of those identified as Windows, 433 (96%) used an unprivileged source port, consistent with Windows DNS behavior. `p0f` also identified 160 (66%) of the DNS resolvers with source port range between 1 and 200 (Section 5.2.3) as Windows systems.

5.3.2 OS-Specific Source Port Ranges. We now use the range of source ports used by a given resolver to identify OS. The pools from which many DNS resolver software implementations select ephemeral ports are, in some cases, specific to OS. If we know the size of the pool of ephemeral ports for a given OS, we can determine from the range of even 10 randomly-selected ports whether the associated packets originated from the OS. The probability distribution of range values for a given pool size can be modeled using a Beta distribution. Given 10 queries from a resolver selecting its source ports uniformly from the pool, the probability distribution is $Beta(\alpha, \beta)$ with $\alpha = 9$ and $\beta = 2$. This distribution tells us how likely it is that a given observed range comes from a pool of a given size.

We experimented with various OSes and DNS software implementations to observe the pools used for source port selection and their fit against our model. We installed BIND 9.11 on the following OSes: FreeBSD 11.3, 12.0, and 12.1; and Ubuntu Linux 10.04, 12.04, 14.04, 16.04, 18.04, 19.04, and 19.10 (Linux kernels 2.6 through 5.3). BIND 9.11 was used because it allocates source ports from the OS-designated ephemeral port range (see Section 5.3.3). We also enabled Windows DNS on Windows Server versions 2003, 2003 R2, 2008, 2008 R2, 2012, 2012 R2, 2016, and 2019. We configured each DNS server to handle recursive queries and issued 10,000 queries with unique query names to each. We then observed the source ports associated with the queries leaving each server instance for the authoritative DNS server, which was also in our lab.

The results were as follows. All Linux kernel versions selected source ports randomly from the contiguous set of values 32768 through 61000—a pool of size 28,232. All versions of FreeBSD consistently used the ephemeral port range designated by the Internet Assigned Numbers Authority (IANA), ports 49152 to 65535—a pool of size 16,383 [25]. Windows DNS (2008 R2 and later) consistently allocated ports from a pool of size 2,500, always within the IANA range, but there were three caveats. First, the start and end values of the pool were determined at server startup and thus were different across running instances. Second, the pool consisted of a contiguous set of ports, with one exception: if the pool started in the highest 2,499 ports of the IANA range, then it wrapped to the bottom of the IANA range. Finally, we observed that BIND 9.11 installed on Windows Server (post 2008) selects from the full range of unprivileged ports (i.e., 1024–65535).

We mention two other points with regard to our analysis. First, only the OSes that allocated ephemeral source ports randomly could be applied to our model. Because Windows DNS 2003, 2003 R2, and 2008 all used a single source port, they were not included in our analysis. Second, we note the discrepancy between the size of ephemeral port pool used by BIND 9.11 (64,511) and that used by Windows DNS (2,500) when each is installed on Windows Server. Thus, we can only uniquely identify Windows Server when it is running Windows DNS software.

To simulate the 10 follow-up queries used to calculate the source port range for the Internet resolvers, we divided the 10,000 queries from each DNS resolver implementation into samples of size 10. We then calculated the range of the source ports for each of the samples, yielding 1,000 sample ranges for each DNS software. However, range calculation for ports from Windows systems needed special consideration. If the pool appropriated by a running Windows DNS server instance "wrapped" around the maximum value and was thus split across the high and low values of the IANA range, it was possible for the 10 ports in a sample to be divided amongst the two non-contiguous parts of the pool. In this case, the computed range of the 10-port sample would be nearly 14,000 higher than what it would otherwise be.

We applied the following algorithm to adjust the ports from Windows DNS to make the range comparable. Let $s = 2500$ represent the size of the Windows DNS port pool. Let $i_{min} = 49152$ and $i_{max} = 65535$ represent the minimum and maximum port values in the IANA pool, respectively. Let $R_{low} = [i_{min}, i_{min} + s - 1)$ and $R_{high} = (i_{max} - (s - 1), i_{max}]$ represent the low and high wrap regions of the IANA range. Finally, let P_n represent the set of source ports observed from resolver r. An adjustment was made to the ports in P_n if: 1) all ports in P_n were either in R_{low} or R_{high}; 2) at least *one* of P_n was in R_{low}; and 3) at least *one* of P_n was in R_{high}. If all of these conditions hold, then for all ports in P_n that were also in R_{low}, the port value was increased by $i_{max} - i_{min}$. This adjustment effectively let ports split across a high range and a low range to be considered as if the range were contiguous. This adjustment is reflected in Figure 2, Figure 3a, and Figure 3b.

Figure 3a shows a histogram of the ranges observed for the 10-query samples that were part of our controlled experiment. In addition to using the samples from the three OSes mentioned from our experiment, we included the results of a DNS server configuration in which ports were selected from a pool spanning 1023 through 65535; this is labeled "Full Port Range." The Beta distribution curves corresponding to the pool size for each OS overlay the histogram to visually demonstrate the match between theoretical and empirical results. Each curve was shifted by the minimum observed range from a given pool and scaled by the difference between the maximum and the minimum observed ranges from a given pool. The different sizes of ephemeral port pools are clearly visible as peaks in the histogram. The tight fit between the histogram and the theoretical Beta curves indicates a strong alignment between the empirical data and the model.

Each row in Table 4 shows a low and a high value for the source port range, three of which are labeled with an OS, per our model and experimentation. The port range cutoff between FreeBSD and Linux (16,331) was optimized to minimize classification error, such that only 0.05% of FreeBSD and 3.5% Linux systems would be misclassified. Similarly, the port range cutoff between Linux and "Full Port Range" (28,222) was optimized such that only 0.35% of those collective systems would be misclassified. All other range cutoffs were selected to achieve 99.9% classification accuracy.

We now apply the port range cutoffs from our model to port ranges observed in connection with our follow-up queries. We adjusted the port values for resolvers that p0f identified as Windows, according to the algorithm described previously. The resulting histogram is shown in Figure 3b, with the Beta distribution curves overlaying the empirical results. The trends in range are clearly identifiable as peaks in the histogram, just as they were in the results of controlled experiment (Figure 3a).

Table 4 shows the number of DNS resolvers at target IP addresses that fall between the source port range cutoffs associated with each OS (i.e., the area under each Beta curve). Using this breakdown, 13,692 (4.6%) were identified as having an ephemeral source port range matching that of Windows Server. While p0f was unable to classify a large fraction of DNS resolvers, it provided additional confidence in our identification of Windows systems. Of the resolvers identified by source port range as Windows, 12,118 (89%) were also identified by p0f as Windows. Port ranges matching pool sizes for

	Reachable IP Targets				
Source Port Range (OS)	**Total**	**Status**		p0f	
		Open	**Closed**	**Win**	**Lin**
0	3,810	1,566	2,244	451	88
1–200	244	201	43	160	17
201–940	144	100	44	76	4
941–2,488 (**Windows DNS**)	13,692	12,179	1,513	12,118	8
2,489–6,124	366	257	109	229	8
6,125–16,331 (**FreeBSD**)	11,462	1,161	10,301	324	91
16,332–28,222 (**Linux**)	89,495	2,430	87,065	134	677
28,223–65,536 (**Full Port Range**)	178,773	11,845	166,928	2,521	6,519

Table 4: Reachable IP targets (IPv4 and IPv6 combined) broken down by observed source port range, open or closed status, and p0f classification. The OS is shown for ranges that correspond to OS port ranges, using a minimum misclassification rate.

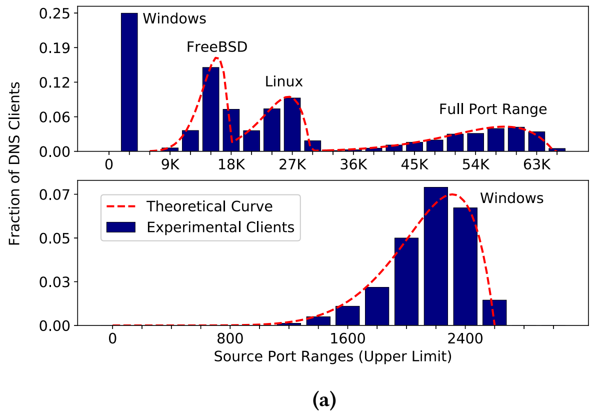
Linux and FreeBSD accounted for 89,495 (30%) and 11,462 (3.8%) of resolvers, respectively.

Table 4 shows that nearly half of the resolvers with a source port range of 0 were open—exploitable even without spoofing! However, considering the 1,802 ASes affected by vulnerable resolver configuration, 1,708 (95%) included at least one closed resolver. For these networks, the lack of DSAV exposes the vulnerability of an otherwise unreachable system.

Remarkably, the largest concentration of open resolvers were categorized as Windows OS, both by source port range and p0f. In fact, considering the overall 13,692 resolvers that were identified as Windows systems by source port range, 12,179 (89%) were open and just 1,513 (11%) were closed! The exact cause of this high correlation is unclear to us, but we did note that the default configuration for Windows DNS 2012, 2016, and 2019 was "open." Additionally, the occurrence is not isolated; Windows systems (by source port range) running open resolvers were found to be in 3,063 ASes—5.0% of all those tested.

5.3.3 Full Port Range. While the "Full Port Range" category of resolvers exhibits the best security posture—in terms of randomness—it leaves a void in our analysis. Possible explanations include 1) OSes other than those examined in this work; and 2) DNS software that uses a pool of ephemeral ports other than the OS defaults.

To further examine DNS software-specific behavior with respect to source port allocation, we installed the following DNS resolver implementations on our instance of Ubuntu 19.10: BIND versions 9.5.0, 9.5.2, 9.6.3, 9.7.7, 9.8.8, 9.9.13, 9.10.8, 9.11.16, 9.12.4, 9.13.7, 9.14.11, 9.15.8, and 9.16.0 (the latest release for each major version); Knot Resolver version 3.2.1; Unbound version 1.9.0; and PowerDNS Recursor version 4.2.0. We issued 10,000 recursive DNS queries to each software installation and examined the source ports that were used. There were three general trends: 1) random selection from the default pool designated by the OS; 2) random selection from the full range of unprivileged ports (i.e., 1024 - 65535); and 3) use of



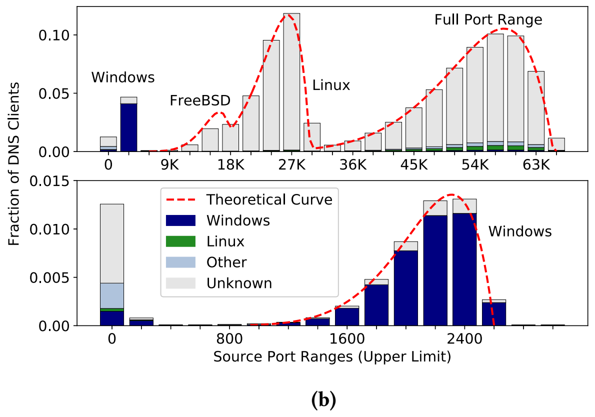


Figure 3: Frequency distribution of the range of source ports used by (a) DNS clients in a controlled lab environment and (b) resolvers reachable with spoofed-source query. The upper plot for each shows the frequency of ranges between 0 and 65,535 (max), and the lower plot shows the frequency of ranges 0 through 3,000 (i.e., a zoomed-in version of the upper plot). The curves overlaying the histograms represent the theoretical models associated with the port range distributions of OS defaults and are labeled with the corresponding OS. The breakdown of OS, as identified by `p0f`, is shown in the composition of each bar in (b).

Software	Source Port Pool (default)
BIND 9.5.0	8 ports, selected at startup
BIND 9.5.2–9.8.8	1024–65535
BIND 9.9.13–9.16.0	OS defaults
Knot Resolver 3.2.1	OS defaults
Unbound 1.9.0	1024–65535
PowerDNS Rec. 4.2.0	1024–65535
Windows DNS 2003, 2003 R2, 2008	1 port, > 1023, selected at startup
Windows DNS 2008 R2, 2012, 2012 R2, 2016, 2019	2,500 contigious ports (with wrapping), selected at startup

Table 5: Default source port allocation behaviors by DNS software.

a single source port or random selection from a small set of source ports. The full breakdown is summarized in Table 5. If we relate these empirical findings back to our analysis of reachable resolvers, the "Full Port Range" part of the histograms in Figure 3b might be any of the DNS resolver implementations that use the maximum source port range, like BIND 9.5.2. We cannot empirically narrow down software version or OS any more succinctly when the full range of unprivileged ports is observed.

5.4 Forwarding

Further surveillance of an internal system allows us to trivially determine if it issues queries to DNS authoritative servers directly, or whether it forwards its queries to upstream DNS resolvers. This might add value to an adversary in the case the upstream networks might be vulnerable. We made this assessment by comparing the source IP address (i.e., the client) querying our authoritative DNS servers against the target IP address embedded in the query name. Because the DNS zone associated with the query names used in our experiment was dual-stack, we relied on the IPv4- and IPv6-only queries (Section 3.5) to make this comparison.

Of the 506,822 IPv4 and 47,978 IPv6 resolvers that resolved our follow-up queries, 269,509 (53%) of IPv4 and 40,631 (85%) of IPv6 addresses queried our authoritative servers directly; 240,491 (47%) of IPv4 targets and 7,566 (16%) of IPv6 targets forwarded our queries to a different address. We note that 3,178 IPv4 and 219 IPv6 targets were found to be in both categories—that is, they forwarded at least one query and issued at least one query directly.

5.5 Local System Infiltration

In addition the knowledge we gained relating to lack of DSAV at networks, we learned something about internal systems (and their OSes) that were similarly vulnerable. Two of the spoofed sources used in our experiment were sources that should never originate from outside the system that receives them: destination-as-source and loopback (see Section 3.3). Considering IPv4 and IPv6 collectively, 123,592 IP targets were reached by destination-as-source queries, and 107 were reached by loopback queries (see Table 3). Using the OS instances and DNS server installations used in Section 5.3, we tested whether these spoofed-source queries reached the DNS service running in user space. Our findings are summarized in Table 6.

Destination-as-source packets were accepted universally by *all* OSes! For Linux-based OSes, only the IPv6 variant were accepted, but for all others, both IPv4 and IPv6 destination-as-source packets were accepted.

Only two OSes accepted queries with loopback as a source. Windows Server 2003 and 2003 R2 accepted IPv4 loopback, and Linux

OS	Linux Kernel	IPv4		IPv6	
		DS	LB	DS	LB
Ubuntu 16.04, 18.04, 19.04	4.15, 5.3, 5.0			●	
Ubuntu 10.04, 12.04, 14.04	2.6, 3.13, 4.4			●	●
FreeBSD 12.1, 12.0, 11.3	N/A	●		●	
Windows Server 2008, 2008 R2 2012, 2012 R2,2016, 2019	N/A N/A	●		●	
Windows Server 2003, 2003 R2	N/A	●	●	●	

Table 6: OS versions and their acceptance of spoofed-source packets, either destination-as-source (DS) or loopback (LB).

accepted IPv6 loopback queries. We reached out to the two operators responsible for 28 (26%) of the resolvers that handled loopback queries. Because they were all IPv6 loopback sources, we suspected that they were running Linux kernel 4.x or earlier. Both responded to our communication and confirmed our suspicions: one organization's servers were running version 3.10 of the Linux kernel and the other's were running version 2.6.

6 DISCUSSION

The findings in this paper are non-trivial. We have shown that, in many cases, systems thought to be accessible only by trusted parties can be reached with a minimal amount of effort. We have also seen that it is not simply the reachability of these systems that matters, but the fact that a potentially malicious third party can identify their OSes and discover their weaknesses with just a few strategically-formed queries. Finally, we observed both through active measurement and through anecdotal evidence—supported by communications with DNS operators—that there is some complacency with regard to security and maintenance of internal systems. These messages collectively convey a message that security at network borders is—in many instances—false, that old, vulnerable software and configurations yet have a deployment presence in the wild, and that substantial effort will be required to motivate the changes necessary to fix these insecurities. Without such change, internal systems continue to be reachable and potentially more vulnerable to various attacks, including DNS cache poisoning (Section 5.2), DNS zone poisoning [29] and NXNS exploitation [43].

Our goal with the current research is not merely to identify the problems with networks and systems, but to spark impetus for widespread change. Increased OSAV adoption (i.e., BCP 38) is certainly part of the solution to source address spoofing in the wild—to prevent both reflection attacks and spoofed-source infiltration. However, OSAV requires the participation of third party networks—those hosting the attackers, not the victims. On the other hand, potential victims of spoofed-source network infiltration can prevent such attacks by configuring their *own* systems for DSAV. This includes both routers at the network border and local systems. Routers should drop packets bearing an internal source address, if they arrive on an external interface. Kernels should drop packets bearing a source address corresponding to any address configured on their system, including loopback addresses. While there might be some legitimate purpose for this behavior, its demand is certainly minimal, and it should not be the default. Even so, every OS that we analyzed allowed IPv6 destination-as-source packets to be received, and all but Ubuntu (Linux) allowed the IPv4 equivalent.

Because of the broad impact of our findings, spreading awareness is important, particularly in connection with the publication of this paper. Our plans include three parts: individual reach-out, broader communications, and testing tools. Using contact information from the DNS (see examples in Section 5.2), WHOIS databases, Regional Internet Registries, and other resources at our disposal, we have initiated reach out to the technical and administrative contacts at affected organizations, beginning with those that show the most vulnerability (e.g., the systems with little or no source port randomization). A broader audience will be reached via operator meetings and conferences, such as RIPE [40] and NANOG [37], where talks and tutorials can be given. Finally, we plan to make the analysis of a network or system available to the general public via a Web interface—both for their testing and our further data collection.

7 CONCLUSION

In this paper we have investigated an area previously explored very little—the lack of DSAV in networks and systems. We presented a methodology for effectively identifying networks and systems vulnerable to spoofed-source infiltration. In November and December of 2019 we sent benign, spoofed-source DNS queries to almost 12 million IPv4 addresses and 800,000 IPv6 addresses in nearly 54,000 and 8,000 respective ASes. Our analysis found that about 5% of IPv4 and 6% of IPv6 queries reached and were handled by their intended targets. However, about half of IPv4 and IPv6 ASNs were infiltrated using our technique. By analyzing the behavior of these resolvers in response to additional spoofed-source DNS queries, we were able to identify the OS of many of the systems we reached behind closed networks. Finally, we identified vulnerable DNS software and systems by analyzing the source port allocation strategies employed by reachable systems. Nearly 4,000 DNS resolvers were found to exhibit no variance in source port across their DNS queries!

The findings in this paper are significant and can have real impact on Internet security. It is our hope that the results of this study, as well as our efforts to encourage change can make that impact a positive one, providing a stronger defense against spoofing, infiltration, and cache poisoning.

ACKNOWLEDGEMENTS

We gratefully acknowledge the Comcast Innovation Fund for their support of the work that produced this material, DNS-OARC for providing access to the 2018 and 2019 DITL collection, and CAIDA for providing us access to their infrastructure to carry out our experiment. We also acknowledge with gratitude the following individuals who provided helpful reviews of our work: kc claffy, Matthew Luckie, and Daniel Zappala. Finally, we thank the IMC 2020 reviewers and our shepherd, Alex Snoeren, for their comments and guidance.

REFERENCES

[1] Baidu. 2020. Baidu. http://www.baidu.com/

[2] Robert Beverly, Arthur Berger, Young Hyun, and k claffy. 2009. Understanding the Efficacy of Deployed Internet Source Address Validation Filtering. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement* (Chicago, Illinois,

USA) *(IMC '09)*. Association for Computing Machinery, New York, NY, USA, 356–369. https://doi.org/10.1145/1644893.1644936
[3] S. Bortzmeyer. 2016. RFC 7816: DNS Query Name Minimisation to Improve Privacy.
[4] S. Bortzmeyer and S. Huque. 2016. RFC 8020: NXDOMAIN: There Really Is Nothing Underneath.
[5] CAIDA. 2020. Spoofer. https://www.caida.org/projects/spoofer/
[6] B. Carpenter and S. Brim. 2002. RFC 3234: Middleboxes: Taxonomy and Issues.
[7] CenturyLink. 2020. CenturyLink Domain Name Server (DNS). https://www.centurylink.com/home/help/internet/dns.html
[8] Cisco. 2020. OpenDNS. https://www.opendns.com/
[9] M. Cotton, L. Vegoda, Ed. R. Bonica, and B. Haberman. 2013. RFC 6890: Special-Purpose IP Address Registries.
[10] J. Damas. 2008. RFC 5358: Preventing Use of Recursive Nameservers in Reflector Attacks.
[11] K. Davies. 2008. DNS Cache Poisoning Vulnerability: Explanation and Remedies.
[12] C. Deccio, D. Argueta, and J. Demke. 2019. A Quantitative Study of the Deployment of DNS Rate Limiting. In *International Conference on Computing, Networking and Communications (ICNC 2019)*. IEEE, New York, NY, USA, 442–447.
[13] J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, and D. Wessels. 2016. RFC 7766: DNS Transport over TCP - Implementation Requirements.
[14] D. Dittrich and E. Kenneally. 2012. *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Technical Report. U.S. Department of Homeland Security.
[15] DNS Operations, Analysis, and Research Center (DNS-OARC). 2018. 2018 DITL Data. https://www.dns-oarc.net/oarc/data/ditl/2018
[16] DNS Operations, Analysis, and Research Center (DNS-OARC). 2019. 2019 DITL Data. https://www.dns-oarc.net/oarc/data/ditl/2019
[17] Domain Name System Operation, Analysis, and Research Center. 2020. DNS-OARC. https://www.dns-oarc.net/
[18] Chad Dougherty. 2008. Multiple DNS implementations vulnerable to cache poisoning. https://www.kb.cert.org/vuls/id/800113/
[19] D. Eastlake and R. van Mook. 2009. RFC 5452: Measures for Making DNS More Resilient against Forged Answers.
[20] P. Ferguson and D. Senie. 2000. BCP 38: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing.
[21] Oliver Gasser, Quirin Scheitle, Pawel Foremski, Qasim Lone, Maciej Korczynski, Stephen D. Strowes, Luuk Hendriks, and Georg Carle. 2018. Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists. In *Proceedings of the 2018 Internet Measurement Conference* (Boston, MA, USA). ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3278532.3278564
[22] Google. 2020. Google Public DNS. https://developers.google.com/speed/public-dns/
[23] Olafur Gudmundsson. 2018. Introducing DNS Resolver, 1.1.1.1 (not a joke). https://blog.cloudflare.com/dns-resolver-1-1-1-1/
[24] H. Marshall Jarrett and Michael W. Bailie. 2015. Prosecuting Computer Crimes. https://www.justice.gov/sites/default/files/criminal-ccips/legacy/2015/01/14/ccmanual.pdf
[25] Internet Assigned Numbers Authority. 2020. Service Name and Transport Protocol Port Number Registry. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
[26] Lamont Jones. 2008. fix query-source comment in default install. https://salsa.debian.org/dns-team/bind9/commit/ed511a4a1182d4434d6c18b33201ae92d1bbb72f
[27] Dan Kaminsky. 2008. Black Ops 2008: It's The End Of The Cache As We Know It, Or: '64K Should Be Good Enough For Anyone'. https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf
[28] S. Kitterman. 2014. RFC 7208: Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1.
[29] Maciej Korczynski, Michał Król, and Michel van Eeten. 2016. Zone Poisoning: The How and Where of Non-Secure DNS Dynamic Updates. In *Proceedings of the 2016 Internet Measurement Conference* (Santa Monica, California, USA) *(IMC '16)*. Association for Computing Machinery, New York, NY, USA, 271âĂŞ278. https://doi.org/10.1145/2987443.2987477
[30] Maciej Korczyński, Yevheniya Nosyk, Qasim Lone, Marcin Skwarek, Baptiste Jonglez, and Andrzej Duda. 2020. Don't Forget to Lock the Front Door! Inferring the Deployment of Source Address Validation on Inbound Traffic. In *Passive and Active Measurement (PAM) conference (PAM 2020)* (Eugene, OR). ACM, New York, NY, USA, 14 pages.
[31] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. 2015. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *Proceedings of the 2015 Internet Measurement Conference* (Tokyo, Japan) *(IMC '15)*. ACM, New York, NY, USA, 355–368. https://doi.org/10.1145/2815675.2815683
[32] Matthew Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A. Kroll, and k claffy. 2019. Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 465–480. https://doi.org/10.1145/3319535.3354232
[33] D. MacFarland, C. Shue, and A. Kalafut. 2015. Characterizing Optimal DNS Amplification Attacks and Effective Mitigation. In *Passive and Active Measurement: 16th International Conference, Proceedings*. Springer International Publishing, Cham, 15–27. https://doi.org/10.1007/978-3-319-15509-8_2
[34] D. MacFarland, C. Shue, and A. Kalafut. 2017. The Best Bang for the Byte: Characterizing the Potential of DNS Amplification Attacks. *Computer Networks* 116 (April 2017), 12–21.
[35] MaxMind. 2020. MaxMind GeoLite2 data. https://www.maxmind.com/
[36] Microsoft. 2020. CVE-2020-1350 | Windows DNS Server Remote Code Execution Vulnerability. https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2020-1350
[37] NANOG. 2020. North American Network Operators Group. https://www.nanog.org/
[38] Jeman Park, Aminollah Khormali, Manar Mohaisen, and Aziz Mohaisen. 2019. Where Are You Taking Me? Behavioral Analysis of Open DNS Resolvers. In *The 49th IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, Portland, OR, USA, 12 pages.
[39] Quad9. 2020. Quad9. https://www.quad9.net/
[40] RIPE NCC. 2020. RIPE Network Coordination Centre. https://www.ripe.net/
[41] Root Server Operators. 2019. Root DNS. http://root-servers.org/
[42] Sarah Scheffler, Sean Smith, Yossi Gilad, and Sharon Goldberg. 2018. The Unintended Consequences of Email Spam Prevention. In *Passive and Active Measurement*. Springer International Publishing, New York, NY, USA, 158–169.
[43] Lior Shafir, Yehuda Afek, and Anat Bremler-Barr. 2020. NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 631–648.
[44] R. van Rijswijk-Deij A. Sperotto and A. Pras. 2014. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In *Proceedings of the 2014 Conference on Internet Measurement (IMC '14)*. ACM, New York, NY, USA, 449–460. https://doi.org/10.1145/2663716.2663731
[45] Verisign. 2020. Verisign Public DNS. https://www.verisign.com/en_US/security-services/public-dns/index.xhtml
[46] P. Vixie. 2013. On the Time Value of Security Features in DNS. http://www.circleid.com/posts/20130913_on_the_time_value_of_security_features_in_dns/.