Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy

Jakub Czyz*, Matthew Luckie†, Mark Allman‡, and Michael Bailey§
*University of Michigan and QuadMetrics, Inc.; jczyz@umich.edu
†University of Waikato; mjl@wand.net.nz
‡International Computer Science Institute; mallman@icir.org
§University of Illinois at Urbana-Champaign; mdbailey@illinois.edu

***Abstract*—There is growing operational awareness of the challenges in securely operating IPv6 networks. Through a measurement study of 520,000 dual-stack servers and 25,000 dual-stack routers, we examine the extent to which security policy codified in IPv4 has also been deployed in IPv6. We find several high-value target applications with a comparatively open security policy in IPv6 including: (*i*) SSH, Telnet, SNMP, are more than twice as open on routers in IPv6 as they are in IPv4; (*ii*) nearly half of routers with BGP open were only open in IPv6; and (*iii*) in the server dataset, SNMP was twice as open in IPv6 as in IPv4. We conduct a detailed study of where port blocking policy is being applied and find that protocol openness discrepancies are consistent within network boundaries, suggesting a systemic failure in organizations to deploy consistent security policy. We successfully communicate our findings with twelve network operators and all twelve confirm that the relative openness was unintentional. Ten of the twelve immediately moved to deploy a congruent IPv6 security policy, reflecting real operational concern. Finally, we revisit the belief that the security impact of this comparative openness in IPv6 is mitigated by the infeasibility of IPv6 network-wide scanning—we find that, for both of our datasets, host addressing practices make discovering these high-value hosts feasible by scanning alone. To help operators accurately measure their own IPv6 security posture, we make our probing system publicly available.**

I. Introduction

Historically, IPv4 has dominated IPv6 in absolute terms, and security issues arising in IPv6 have seemed no more than a small-scale annoyance. However, the character and amount of IPv6 in recent years is changing [17]. For example, Google reports 8% of its users accessed their services over IPv6 in mid 2015—a statistic that has doubled in each of the last five years [29]. Further, many large networks now report double-digit IPv6 deployment percentages (e.g., Comcast 39%, ATT 52%, Deutsche Telekom 28%) [6]. Thus, any threat posed by IPv6 looms large, and, as a community, we should aim to understand and mitigate such threats as early in the adoption process at possible.

NDSS '16, 21-24 February 2016, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2016.23047

With IPv6 connectivity come two forms of security issues. The first type are the set of completely new vulnerabilities that stem from IPv6 protocol changes and features (e.g., [27], [51]). Part of the problem for vendors and operators alike is that there are nontrivial technical hurdles to fully supporting IPv6, especially in policy devices, such as firewalls and IDSes. For instance, due to the flexibility of IPv6 header chains and issues related to fragmentation, simple stateless firewalls, which may have sufficed for IPv4, are not appropriate in IPv6 [14], [19]. While this issue was addressed in a standards document that requires the first fragment to contain the transport headers [27], it will likely take time for compatible hardware and practices to permeate the network. There are a number of such deficiencies or ambiguities in the IPv6 specification that may leave dual-stacked networks vulnerable to emerging attacks via carefully crafted packets.

In this paper, we tackle the second type of security issue that results directly from the dual-stack nature of IPv6 adoption. Quoting IETF IPv6 operational guidance (emphasis is our own):

> From an operational security perspective, [dual-stack] means that you have twice the exposure. One needs to think about protecting both protocols now. At a minimum, the IPv6 portion of a dual stacked network should maintain parity with IPv4 from a security policy point of view... given the end-to-end connectivity that IPv6 provides, it is also recommended that hosts be fortified against threats... There are many environments which rely too much on the network infrastructure to disallow malicious traffic to get access to critical hosts. *In new IPv6 deployments it has been common to see IPv6 traffic enabled but none of the typical access control mechanisms enabled for IPv6 device access.* With the possibility of network device configuration mistakes and the growth of IPv6 in the overall Internet, it is important to ensure that all individual devices are hardened against miscreant behavior. [15].

Specifically, we seek to study how well-worn operating system, protocol, and application weaknesses may be exploitable via the IPv6 network because security policy that has been codified for IPv4, has been neglected in IPv6.

Given the above sketch and standards recommendation, our expectation is that operators' intent is parity across IPv4 and IPv6 security policy in dual-stack settings. While intent is impossible to derive in the large and there are no doubt cases

where operators intentionally set different security policies for the two network layer protocols, we believe these are likely a small minority of the cases. Our contacts with operators, described in § VII bear this out. Therefore, we assume intended parity between IPv4 and IPv6 and we test for lack of parity for various application ports. In this study, we hypothesize that there will be discrepancies due to several factors: (*i*) the lack of a full suite of ancillary tools and software for IPv6; (*ii*) less operator experience with IPv6; and (*iii*) the ability of IPv6 nodes to auto-configure without end-user intervention. More specifically, we hypothesize that ports are blocked less frequently when access comes via IPv6 compared with IPv4. In our experiments, we find many cases that bear this out. Mindful that we may mis-read intent, discrepancies at the very least point to places with additional attack surface.

We examine filtering policy as it applies to dual-stack hosts identified through DNS—those with both IPv4 and IPv6 records. In total, we test filtering policy for 25K dual-stack routers (11 ports/applications) and 520K dual-stack servers (19 ports/applications). We find that:

- **IPv6 is more open than IPv4.** A given IPv6 port is nearly always more open than the same port is in IPv4. In particular, routers are twice as reachable over IPv6 for SSH, Telnet, SNMP, and BGP. While openness on IPv6 is not as severe for servers, we still find thousands of hosts open that are only open over IPv6.
- **When policy variances exist, they tend to exist network-wide.** Our analyses of differences between IPv4 and IPv6 policy show these policy differences tend to be consistent within autonomous systems and routed prefixes (e.g., for 78% of routers in the average prefix, their IPv4/IPv6 policy differences are consistent across the entire routed prefix).
- **Not only does policy differ, but policy mechanism as well.** We classify the type of connectivity failure through the addition of traceroute measurements and show that differences exist not only in how policy for various services are handled, but in how IP protocol versions are as well (e.g., 9% more routers respond *actively* over IPv6 when ports are closed, indicating fewer policy devices silently dropping than on IPv4).
- **Existing IPv6 open services are easily discovered through scanning.** Our analysis of host addressing patterns shows that for a given subnet, most routers and a quarter of servers could be discoverable within one second using state of the art port scanners.

We attempt to confirm our results by contacting various network providers and show that the twelve operators we were able to reach confirmed our results, with ten immediately making changes to correct the errors. To assist operators in accurately measuring and assessing their own IPv6 security posture, we make our scanning tool (sc_filterpolicy) publicly available [4]. Together, the results point to much work left to secure IPv6, whose changes and complexity lie far beyond merely a larger address space.

The remainder of this paper is organized as follows. We begin with a description of methodology in § II and explore if our dual-stack identification is effective in § III. In § IV we determine whether IPv4 and IPv6 policy differences exist, to what extent they exist, and whether the differences vary by application. § V explores if these policy differences are applied consistently within a network and represent a unified policy or likely represent more trivial or piecemeal configuration errors. In § VI we then discuss what differences between protocol versions, if any, exist in where and how policy mechanisms are applied. Next, in § VII we confirm our results by contacting various network providers and show that the twelve network operators that we were able to reach confirmed our results. Finally, in § VIII we revisit the idea that impact of an increased attack surface is mitigated by the infeasibility of IPv6 network-wide scanning, discuss related work in § IX, and conclude in § X.

II. Methodology

A. Developing Target Lists

To explore potential policy discrepancies between IPv4 and IPv6 we first must find dual-stack hosts such that we can measure application reachability (i.e., connection success) to the same target via the two protocols. As we describe below, we harvest lists of IP addresses and host names to start the process. Given a set of hosts, our strategy is to use DNS names as the basic connective tissue between hostnames, IPv4 addresses, and IPv6 addresses. One reason for this approach is that "DNS is one of the main anchors in an enterprise [IPv6] deployment, since most end hosts decide whether or not to use IPv6 depending on the presence of IPv6 AAAA records..." [14]. Each host H^x in our dataset contains three sets of labels that represent H^x: H^x_N is the set of names, H^x_4 is a set of IPv4 addresses and H^x_6 is a set of IPv6 addresses. After obtaining an initial list of hosts (see below) we then detect common elements across multiple hosts. We compare hosts pair-wise, as follows:

$$s = (H^x_N \cap H^y_N) \cup (H^x_4 \cap H^y_4) \cup (H^x_6 \cap H^y_6) \quad (1)$$

When $s \neq \emptyset$ there is overlap between the two hosts and therefore we replace H^x and H^y with a new H^z that represents the merger of the two original hosts, as follows:

$$H^z_N = H^x_N \cup H^y_N \quad (2)$$
$$H^z_4 = H^x_4 \cup H^y_4 \quad (3)$$
$$H^z_6 = H^x_6 \cup H^y_6 \quad (4)$$

Finally, we prune hosts that do not have at least one valid and routable IPv4 and IPv6 address.

There is a possible concern with using DNS as the connective tissue. Labels within a group could actually represent multiple distinct machines, not a single dual-stack machine [12]. As part of our probing we obtain host signatures (e.g., an *SSH* host key). In roughly 3% of hosts we detect multiple signatures across different host labels. We believe this is a strong indication that our process is predominantly identifying dual-stack hosts, and we discuss this point further in § III.

There is also a possible issue with using DNS names as the *source* of targets for our probes in the first place. Because we start from hosts that have names in DNS (more precisely, that have associated sets of A, AAAA, and PTR records), we

will naturally exclude hosts without DNS presence or without complete record sets. We acknowledge that this may bias our results. However, we believe this bias would result in an *underestimate* of the core policy misalignment finding. First, without A or AAAA records, server hosts could not be reached by most clients (except when addresses are directly hard-coded in applications, which is uncommon). Routers could still be functional without these records, but ease of management and configuration dictates that interfaces are often named. Further, for PTR records, maintaining these is a known best practice [10], and, in the case of some servers, may be necessary (e.g., for sending mail [34]). Thus, because maintaining all three record types is generally a sign of correctly operating an Internet-facing service interface (i.e., public router or server), we believe results for these targets will likely produce a lower bound of actual security policy misalignment in the overall dual-stacked server and router population, as the larger population will include less properly-operated hosts.

As we note above, we start the process of identifying dual-stack hosts using two different lists of hosts, as follows.

Router List: Our first focus is on Internet routers, which form the core of the network, and, hence, correct application of security measures is crucial. An attacker that can compromise a router might be able to redirect traffic for man-in-the-middle attacks, cause a complete or targeted network outage, or adjust filtering rules on the router that otherwise serve to protect the network from other attacks. Our router dataset is derived from router interface IP addresses found in Internet-wide *traceroute* data taken by CAIDA's Ark measurement platform [13]. Each system in the Ark platform infers the forward IP path with *traceroute* measurements to random IPv4 addresses in every routed /24 prefix, as well as to the first (::1) and a random address in every routed IPv6 prefix. We extracted the source IP addresses of ICMP hop-limit responses (intermediate routers) archived during December 2014 and performed reverse DNS lookups to obtain names where available. This forms the initial list of routers, which we then refine using the procedure above.

Server List: Our second list consists of servers, which make specific services widely available over the Internet. The security posture of servers can be of significant importance, as servers can hold private information, infect many clients, and strengthen botnets. We initialize our list with the Rapid7 DNS ANY datasets from the `scans.io` repository [47]. This dataset represents DNS ANY queries for a set of hostnames gathered from (*i*) more than 200 DNS TLD zone files (including the popular .com and .net to emerging TLDs such as .farm and .toys); (*ii*) reverse DNS names of IPs detected in HTTP scans, SSL certificate scans, and IPv4-wide reverse-DNS lookups; and (*iii*) <a> tags in HTTP responses to HTTP scans. We use the Rapid7 DNS consumer hostname filter [42] in an attempt to remove all hostnames that appear to be associated with static or dynamic customer-premises IPs (e.g., ip-192-168-0-1.example.com), leaving hostnames that do not appear to be automatically-generated for consumer Internet endpoints. Due to the filter and the provenance of the names and addresses in the list, the list could include some dual-stacked clients; however, we believe that these are in the minority. To validate, we examine what fraction of the IPv4 addresses associated with each host is part of known consumer network blocks, using the SpamHaus Policy Block List (PBL) [50]. We find that 97% of the hosts are not in PBL ranges, as expected. The raw DNS dataset represents approximately 1.4B name to address mappings. After culling the list to just A and AAAA records, we detect common names and addresses, as sketched above, to reduce the set to 2.4M addresses associated with 950K dual-stacked hosts.

B. Probing

We perform active probing to assess security posture differences between IPv4 and IPv6. When probing routers, we use services that are some combination of (*i*) likely to be running on routers (e.g., SSH), (*ii*) crucial to router operation (e.g., BGP) and/or (*iii*) problematic when leveraged by an attacker (e.g., NTP [18]). Thus, the exposure of all of these ports generally increases the attack surface of routers. We probe these services on all routers in our dataset:

> ICMP echo, SSH (TCP/22), Telnet (TCP/23), HTTP (TCP/80), BGP (TCP/179), HTTPS (TCP/443), DNS (UDP/53), NTP (UDP/123), SNMPv2 (UDP/161)

Similarly, after developing the list of dual-stack servers, we perform active probing but with a different set of application types that are more apropos for servers than routers, as follows:

> ICMP echo, FTP (TCP/21), SSH (TCP/22), Telnet (TCP/23), HTTP (TCP/80), HTTPS (TCP/443), SMB (TCP/445), MySQL (TCP/3306), RDP (TCP/3389), DNS (UDP/53), NTP (UDP/123), SNMPv2 (UDP/161)

To select these, we consulted literature on prevalence of scanning [21], prevalence of port blocking [33], as well as application DDoS amplification susceptibility [44]. We wanted to minimize probing ports of lower deployment or interest, as well as constrain our set to a small number so as to minimize load on targets. Ultimately, the potential impact of breach was the most important factor for inclusion[1].

We use two probing methods to collect the data we use in the remainder of the paper. "Basic" probing consists of single probe packets to each service via both IPv4 and IPv6. For ICMP this is an echo request, for TCP it is a SYN segment, and for UDP this is an application-specific request (e.g., DNS A query for "www.google.com", NTP version query, or SNMP query for the sysName.0 MIB using the default public community string). We probe every two weeks starting in mid-January 2015 and mid-February 2015 for the routers and servers, respectively, through July 2015. We found little difference between results over time and, therefore, focus on the router collection from February 19, 2015, which we denote $\mathcal{R}_B$ and the server collection from April 10, 2015, denoted $\mathcal{S}_B$. Our second probing strategy is based on *traceroute* style measurements using the same probe types. We also collected a number of these *traceroute* datasets. Again, our analysis shows similar results across time and therefore we concentrate on the router dataset collected on June 5, 2015, which we denote $\mathcal{R}_T$, and the server dataset collected on July 10, 2015, which we denote $\mathcal{S}_T$.

[1] After initial experiments, several applications were showing minuscule dual-stack response rates (generally a tenth of a percent or less). To focus on more prevalent applications, we dropped these from study. They included, for routers: TFTP, and for servers: IPMI, MS-SQL, NetBIOS, SSDP, and VNC. We also excluded SNMPv1, as results closely matched SNMPv2.

We used *scamper* [35], a parallelized bulk probing tool that supports various types of probes, including *ping* and *traceroute* over ICMP, TCP, and UDP, to conduct our measurements for both IPv4 and IPv6. Because there was no implementation of traceroute that considered application-responses (traditional traceroute deliberately chooses high-numbered, unused ports to solicit ICMP port unreachable error messages) we extended scamper's traceroute implementation to record UDP application-level responses. We configured *scamper* to probe the ports listed above using each IPv4 and IPv6 address of every host. To limit the burden on our measurement targets we tested one port at a time and in random order. Our goal in doing so was to remove the possibility that we would trigger rate limiting by probing the host too quickly and thus raise the possibility that a host that was initially responsive would become unresponsive, conflating a rate-induced outage with a policy to discard specific types of packets. We configured scamper to probe at 5000 packets per second; the basic router measurements took approximately eight hours to measure while the basic server measurements took approximately 22 hours. We paused for at least 1 second between measurements to a given host. Therefore, despite a relatively high probing rate, we spread the load across a set of targets so that we had negligible impact on individual hosts measured.

TABLE I: Dataset summary

	Probe Date		Addresses		Hosts	
Dataset	(2015)	Names	IPv4	IPv6	Total	Suitable
$\mathcal{R}_B$	Feb 19th	38K	41K	41K	35K	25K
$\mathcal{R}_T$	Jun 5th	38K	41K	41K	35K	25K
$\mathcal{S}_B$	Apr 10th	8.3M	1.0M	1.4M	947K	520K
$\mathcal{S}_T$	Jul 10th	8.5M	1.0M	1.4M	951K	533K

Table I describes the datasets we use in the remainder of the paper. The "suitable" column represents the hosts we probe and is the set that respond to ICMP echo in both IPv4 and IPv6. We only measure policy congruity on suitable (responsive) hosts to avoid mistaking a completely unreachable host from one where application policy controls are enforced. While this test is not foolproof, we know these hosts are responsive to both IPv4 and IPv6. We could be excluding hosts that apply different ICMP policies to IPv4 and IPv6, as well as those that filter all ICMP requests. However, we believe the set of hosts we leverage is large enough to give us broad insight into the policy differences between IPv4 and IPv6 across the Internet.

Hosts we measure are spread across the network, and encompass 58% of dual-stack ASes observed in public BGP tables (all available Route Views [52] and RIPE RIS [43]) on midnight February 1, 2015. The $\mathcal{R}_T$ target list contains hosts from over 2K routed prefixes, 1K autonomous systems (ASes) and 70 countries. The $\mathcal{S}_T$ target list, on the other hand, contains hosts from over 15K routed prefixes, 5K ASes and 133 countries. Unsurprisingly, while we leverage a breadth of targets, the set is also skewed. In the $\mathcal{R}_T$ list we find that 19 ASes that belong to the ten most-represented network operators in our list account for half the hosts. Similarly, we find that ten ASes belonging to large hosting/content providers make up half the servers.

C. Ethical Considerations of Probing

Research involving active measurement of networks potentially creates ethical issues as both the conduct of such research and the disclosure of results thereof may result in harm to a variety of stakeholders, including, but not limited to: research institutions, service providers, network operators, and end users. We take note that, while the security community has not reached consensus on standards for such research, existing published work in the field [22], as well as broad ethical guidelines [9] provide a roadmap for how one may minimize the potential for such harms. For example, in the conduct of this research we: (*i*) signaled the benign intent of work through WHOIS, DNS, and by providing research details on a website on every probe IP address; (*ii*) significantly rate limited the probes to minimize impact; (*iii*) limited ourselves to regular TCP/UDP connection attempts followed by RFC-compliant protocol handshakes with responsive hosts that never attempt to exploit vulnerabilities, guess passwords, or change configurations; (*iv*) we respect opt out requests and seed our opt out list with previous requests provided to other researchers [22]. In mitigating disclosure harms, we carefully avoid providing target lists in the published result and notify, by email to abuse contacts, the most egregious networks prior to publication, so that they may correct vulnerable configurations.

D. Result Interpretation

One final methodological task involves interpreting the results of the probes. First, we must decide if a probe succeeds or fails. We define success as reception of (*i*) an ICMP echo reply message in response to an ICMP echo request, (*ii*) a TCP SYN+ACK in response to a TCP SYN and (*iii*) a UDP response to a UDP request. We consider anything else—including no response—as connection failure (e.g., ICMP unreachables, a non-SYN+ACK TCP packet). Once we have a decision for probes within some H_4^x, we make a final IPv4 determination based on majority vote across all IPv4 addresses when there are multiple. Likewise for the IPv6 addresses H_6^x.

Lastly, a minor note on terminology: the versions of IP we study (at OSI layer 3) as well as the applications we probe (at layers 5-7) can all be called *protocols*. To avoid confusion, however, we reserve that term for the IP version under study and instead use the term *application* to denote the protocol/application at the higher layers (e.g., SSH, NTP, etc.) for which we are measuring connectivity.

III. Calibration

A core assumption we make is that all the labels we find for some host H^x point to the same host. This is not of only theoretic concern, as previous work shows DNS names mapping to IPv4 and IPv6 addresses do not always point at a dual-stack host but instead to multiple hosts [12]. Since our goal is understanding security posture of dual-stack hosts, we first calibrate our method for aggregating labels into hosts. To test our assumption we seek to collect application-level information for each H^x in $\mathcal{R}_B$ and $\mathcal{S}_B$ via both IPv4 and IPv6, as follows.

HTTP: We send each host that responds to TCP/80 a HEAD request and extract the server version string (including OS) from responses (e.g., "Apache/2.2.22 (Debian)"). While we do not exclude any, the three most frequently returned strings are

TABLE II: Alias validation via application signatures. A majority (96% of $\mathcal{R}_B$ with data and 97% of $\mathcal{S}_B$) of hosts with signature data matched fingerprints among all host address members.

	$\mathcal{R}_B$ List		$\mathcal{S}_B$ List	
Application	Hosts	Same Sig	Hosts	Same Sig
http	269 (1.1%)	97.0%	235,575 (46.2%)	99.2%
https	183 (0.8%)	96.7%	96,468 (18.9%)	94.2%
snmp2c	12 (0.1%)	100%	41 (0.0%)	95.1%
ntp	843 (3.6%)	97.0%	3,462 (0.7%)	99.1%
ssh	603 (2.6%)	96.7%	218,100 (42.8%)	98.9%
mysql	–	–	1,055 (0.2%)	99.5%
Overall	1576 (6.7%)	96.4%	303,111 (59.4%)	97.1%

indistinct as version and OS are not provided (e.g., "Apache" (39%), "nginx" (23%), and "cloudflare-nginx" (6%)). The next 20 most-frequently returned strings are more specific and provide a stronger fingerprint for matching.

HTTPS: We are able to collect an extensive set of information about each host responding on TCP/443 probes, using both the *openssl* client and *NMAP* with the ssl-enum-ciphers.nse script [3], including: (*i*) the supported cipher suites (for all except SSLv2-only hosts); (*ii*) the supported SSL/TLS protocol subset of {SSLv2, SSLv3, TLSv1, TLSv1.1, and TLSv1.2}; (*iii*) the actually negotiated protocol between client and server; and (*iv*) the server's certificate fingerprint.

SNMP: For SNMP, we retrieve two MIBs—*sysDescr.0* and *sysName.0*—via SNMP version 2c *get* requests to the *public* community. Responses include the OS (including version) and an administrator-set system name. We require responses to both gets to complete an identifying fingerprint.

NTP: For each host responding to UDP/123 queries we issue the *version* command using the *ntpq* tool. This provides a semi-structured string containing: *version, processor, system, stratum, precision, refid, reftime, frequency, status,* and *associd* among other fields that we do not further utilize as they are less common or vary across queries–not useful for fingerprinting.

SSH: We use the *ssh-keyscan* utility (part of the OpenSSH-clients tools) to obtain the SSH server version, the key length of the server's encryption key, and the fingerprint of the key.

MySQL: For hosts with open TCP/3306 we send two newline characters causing servers to print an identifying banner, which we harvest (stripping unprintable characters). As noted above, we do not probe MySQL on routers, and, therefore, this fingerprint is only available for the server host list.

For each host x on our host lists we collect the above information via probing to every IP address in H_4^x and H_6^x. We then check for consistent behavior across all applications that respond on all addresses. When we find consistency across all IP addresses for H^x we conclude that the host is highly likely to be a single dual-stack host. Even if this conclusion is wrong, we believe the identical configuration indicates the operator's intention is to provide the same service across IPv4 and IPv6 and therefore policy differences are important to illuminate.

As a bound on our ability to match fingerprints, we first assess the general openness of our targets to analyzing the information we used for consistency checking. If we collapse our basic probe results for each host across applications tested and both IPv4 and IPv6 (i.e., a very coarse-grained analysis of the results we discuss in sections to come), we find that we can access at least one of these signature-providing applications via at least one of a given host's IP addresses for 44% and 76% of the hosts in $\mathcal{R}_B$ and $\mathcal{S}_B$, respectively. In other words, our technique will have no data at all to fingerprint a host for over half the routers and nearly a quarter of servers. In fact, since we need at least one IPv4 and one IPv6 fingerprint for the same port to do matching, the number of hosts we can actually match signatures for across IP versions is even lower (7% and 59%, respectively, as discussed below). However, we stress that we are calibrating our *technique* of aggregating sets of labels (i.e., results for IPs grouped via their associated A, AAAA, and PTR records) into hosts and not each individual assessment. As such, the sample, though biased toward more open hosts, seems sufficiently large to represent such name-based aggregation.

Table II shows the results of our consistency probing. For each host with signature data from at least one signature-providing application open on all associated addresses, we find a high level of consistency—96% for routers and 97% for servers. This roughly agrees with previous work showing that, while it is not exceedingly rare for hostnames with both IPv4 and IPv6 addresses to represent different machines, 93% of the time they in fact do represent the same system [12]. For servers, our results cover nearly 60% of the hosts in the $\mathcal{S}_B$ host list. On the other hand, we can make a signature-based comparison for only roughly 7% of the routers in the $\mathcal{R}_B$ list. While the coverage in $\mathcal{R}_B$ is low, we note that, since routers have less general openness than servers, we expect our consistency check for routers to be useful in fewer cases. Even with the low coverage we believe the high consistency rate validates our methodology for aggregating labels into hosts.

The small fraction of hosts for which a signature match between the IPv6 and IPv4 addresses fails suggests operators using a separate server for IPv6 than for IPv4 behind the hostname. We can speculate as to various explanations for such a configuration, including matching resources to load, where the IPv4 address points to a load balancer, as most traffic is via IPv4, while the IPv6 points to a single specific server that supports IPv6. We do not exclude the hosts without matching signatures nor those failing matches, mainly for the reason that we contend that *the semantics of a single hostname are a single service*. As such, whether it is deployed on one physical machine, configured identically, or neither, Internet users and application routers do not expect that the services available via a hostname differ based on the network protocol used to reach them, just as they do not expect DNS-based load balancing with multiple A records for a name to provide different service depending on the address their host happens to select from the several available. In other words, while the calibration in this section is useful for understanding the underlying population of machines that our hosts represent, the policy misalignment we find is orthogonal to and no less troubling whatever these signature matching results for any name show.

IV. Baseline Policy Discrepancy

Overall, 26% of routers and 26% of servers were reachable in IPv6 for at least one application not reachable in IPv4; five of eight tested applications are more open over IPv6 for

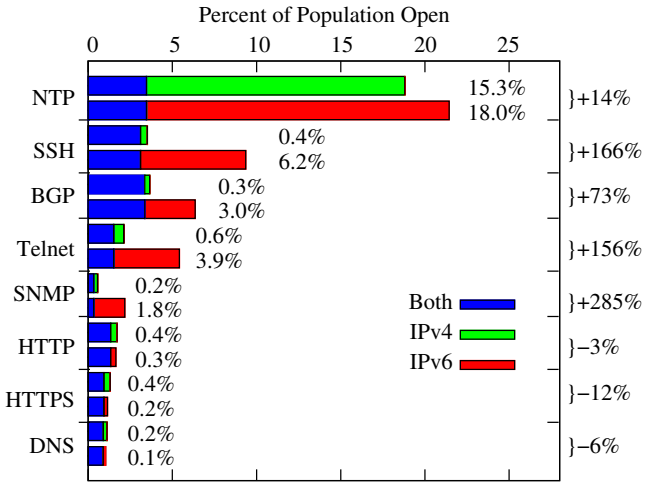


Fig. 1: Percentage of 25K dual-stack routers ($\mathcal{R}_B$) responsive to ping that were open via IPv4 and/or IPv6 for each application tested. For each application, the green bar corresponds to reachability (connection success) over only IPv4, the red bar only IPv6, and the blue bar reachability over both. Beside each bar we report the percentage of hosts tested that were only reachable by IPv4 or IPv6, and beside each application is the percentage difference in reachability over IPv6 compared to IPv4.

the routers, and six of eleven tested applications are more open over IPv6 for servers. While 18% of routers and 17% of servers we tested were reachable in IPv4 for at least one application not reachable in IPv6, some applications can have default configurations that do not listen in IPv6. The policy discrepancy landscape overall is profoundly varied; a staggering 44% of routers and 43% of servers had different application reachability (i.e., connection success) depending on version of IP used. At a high level, this suggests a large difference in services that dual-stacked hosts effectively make available (intentionally or not) over one version of IP versus the other.

A. Router Application Openness Results

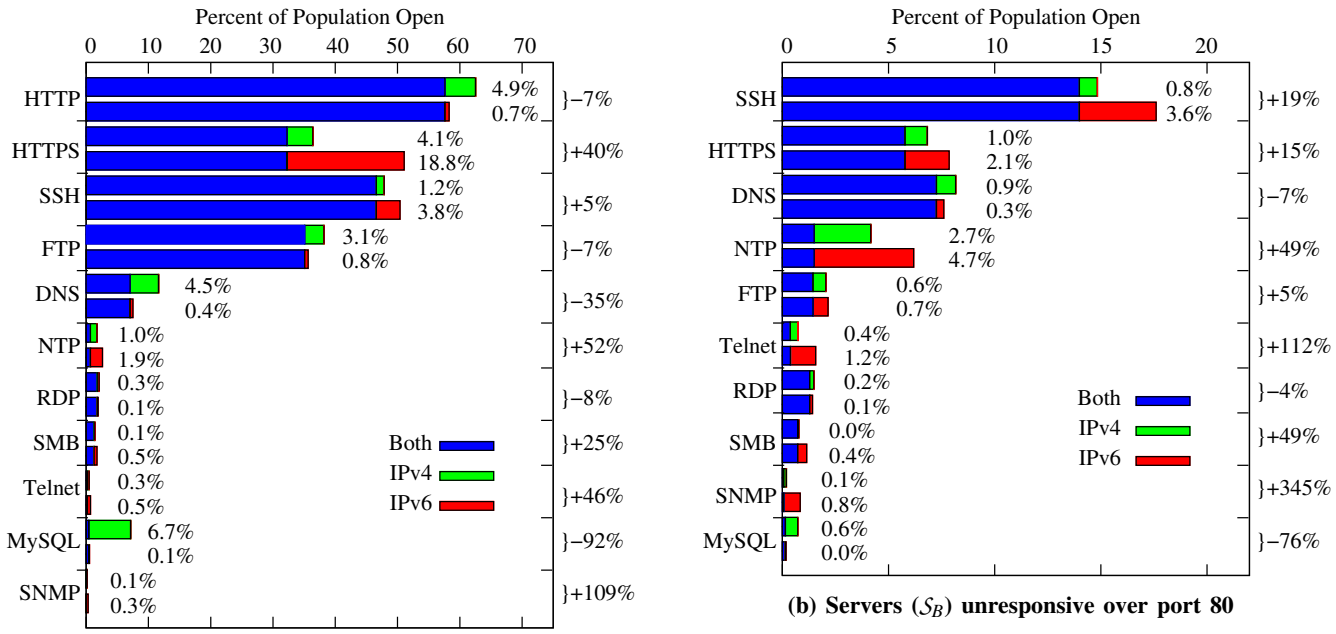
Figure 1 shows the protocol discrepancy observed between IPv4 and IPv6 for routers. For each application, we show the percentage open over IPv4 and/or IPv6, the percentage open over only IPv4 or IPv6, and the difference in openness of IPv6 over IPv4. Particularly troubling is an observation that the three most open protocols in IPv6 are high-value: SSH, BGP, and Telnet; these protocols were 166%, 73%, and 156% more open in IPv6 than IPv4, respectively. We next discuss each application result and comment on its possible impact.

NTP: Among tested applications, NTP is most open overall, but discrepancy between the two protocols is relatively moderate at 14% more openness for IPv6. The fact that NTP is the most reachable application in this dataset is not totally unexpected, given that this application is commonly enabled by default on network devices (e.g., [1]). A surprising finding is that a relatively large percentage of the routers only respond via one protocol or the other relative to those that respond on both. This suggests some peculiarity in default router NTP configurations. While access to NTP is not a critical risk, it has been leveraged for large-scale distributed denial-of-service (DDoS) attacks in the past, and lagging IPv6 protection may signal less attention paid to blocking its traffic over IPv6 than has been deployed for IPv4 [18]. Further, we found that the NTP version command we used can leak the device vendor and version in many cases, which may be helpful to attackers targeting specific vulnerabilities.

SSH: The second most open application we see is SSH. SSH also has the second largest discrepancy between the two protocols, with IPv6 being more than twice as open; 166% more routers allow connecting over IPv6 than IPv4. As SSH is a management application allowing control over the device, this is a troubling finding. If exploited via brute-force password attempts, harvested passwords used by administrators on other compromised sites or hosts, or via software vulnerabilities, SSH access could lead to stealthy and large-scale attacks. As with most router vulnerabilities, these might include, for example, redirecting traffic for specific websites, email, or DNS queries to attackers, and facilitating other various forms of man-in-the-middle attacks. Further, since routers are specialized systems with typically proprietary operating systems and less general-purpose computing power, they may be less likely than servers to be bolstered with protections against a range of SSH-based attacks—e.g., password attempt limits, SSH key-only logins, and logging failed attempts.

BGP: The third most open application is BGP, which we would expect to be running on routers, but not to necessarily be open for anyone to connect. An open BGP port on routers leaves them potentially more susceptible to various TCP-based attacks, such as SYN floods, and blind in-window attacks [36]. The fact that 73% more hosts completed the TCP handshake over IPv6 than IPv4, suggests, at the very least, that some additional protection, likely via an access control list, has been set up on these devices for IPv4 but not for IPv6. Hence, the deployed security policy on these routers for IPv6 contradicts their IPv4 policy. As routers constitute the backbone of the Internet, and BGP is the protocol by which Internet routers communicate where to send traffic, vulnerabilities in BGP pose a serious threat.

Telnet: The fourth most open application is Telnet. We were surprised to discover so many routers accept global TCP connections to Telnet at all (9% of the dataset over any IP version), given the fact that this application has been replaced by SSH as a primary management interface for routers, in large part due to its inherent insecurity. This insecurity stems mostly from the fact that Telnet sends traffic unencrypted and that, unlike SSH, it also has no means of validating the identity of the server that a client connects to (which an SSH client can do by checking the fingerprint of the key that the server provides during connection). Moreover, beyond server authentication, there is no key-based authentication for clients in Telnet either; so, all connections involve sending a user name and password in clear text to a server whose identity can not be verified. Router Telnet sessions have even been targeted by nation states to capture the configuration of routers, leading to deeper network breach [24]. As with SSH, the danger of weak passwords that can be brute-forced and the possibility of shared passwords across sites allowing compromised credentials to be used to gain broader access, mean that the security posture of these devices is degraded simply by having Telnet exposed. As there are again more than double—156% more—IPv6-open routers



(a) Servers (S_B) **(b) Servers (S_B) unresponsive over port 80**

Fig. 2: Percentage of 520K dual-stack servers responsive to ping that were open via IPv4 and/or IPv6 for each application tested (figure 2a) and that of 137K (37% of all) servers that were not responsive to HTTP (figure 2b). Seven of the eleven applications tested are more open in IPv6, including the security-critical SSH, SNMP, SMB, and Telnet services. The subset of servers that are not HTTP servers are more open than the general server population.

with Telnet exposed, the deployment of IPv6 here has markedly reduced security in this sample of routers.

SNMP: We attempted SNMPv2c requests over UDP/161 for the sysName.0 MIB using the common default *public* community string. Three percent of routers responded with data. We did not attempt to use the common *private* community to alter configuration on systems, nor did we collect any data from the device. However, when we did our follow-on probes for signature matching described in § III, we additionally performed SNMP gets for the sysDescr.0 MIB, which allowed us to aggregate operating system versions, the large majority of which reported being Cisco. While the read-only *public* community may itself not necessarily pose a catastrophic risk to the device, it may be used to leak version information, find weaknesses in configuration, or gather information about connected devices. All of this can be useful reconnaissance for attackers, especially when paired with published vulnerabilities. Furthermore, the fact that these devices expose SNMP for nearly four times (285%) more hosts over IPv6 than IPv4 suggests that many operators took steps to block this management application over IPv4. If these operators are relying on access-lists, firewalls, or other port filtering for protection of SNMP but keeping the default community strings in place, it is likely that this population of routers could be reconfigured using the private community over IPv6, a much more direct and immediate threat than that posed by the read-only probe we attempted. As such, we consider this a serious vulnerability.

HTTP and HTTPS: The web protocols were not very common on routers, and are the first to break the pattern of greater IPv6 openness, with each slightly more closed for IPv6 than IPv4 (-3% and -12%, respectively). For routers with web management enabled, this means security is probably no worse under IPv6 than IPv4. There were a small handful of hosts where access was only allowed over IPv6 (78 routers for HTTP and 51 HTTPS), suggesting, perhaps, at least some cases where IPv4 access was blocked but similar blocks were not in place for IPv6. Embedded web-based management applications are notorious for vulnerabilities, and this capability is rarely used by professional router operators. Having an unknown web-based attack vector over IPv6 enabled, even for this small number of hosts is problematic. Fortunately, the scale here is small.

DNS: Like HTTP and HTTPS, DNS was less open for IPv6, though, again, a small handful of routers (35) only responded to DNS over IPv6 and not IPv4. Aside from application-specific vulnerabilities (e.g., BIND CVEs) that might impact the device if DNS is exposed, other notable security implications of having DNS open to the public when policy would dictate otherwise have to do with (*i*) leaking internal-only DNS records and (*ii*) facilitating DDoS attacks using DNS. After HTTP, router openness for the two IP versions was most similar for DNS.

Overall, the baseline protocol differences we found in this population are troubling. The fact that more than a quarter of routers had at least one application accessible over IPv6 that was closed over IPv4, including some high-value application ports for attackers, means that the routers in our sample are generally more vulnerable under IPv6 than IPv4 (at least on the tested common applications). Since network operators are at the forefront of understanding and deploying IPv6, this is somewhat surprising. We conjecture that network hardware may be subject to less security audits and scrutiny than servers are, although it is also possible that, as router operators usually deploy IPv6 (naturally) before the server operators that rely on it do, they may be doing so under either greater time pressure

or with fewer existing institutional tools and processes for assuring consistent security policy on routers.

B. Server Application Openness Results

Figure 2 shows the protocol discrepancy we observed between IPv4 and IPv6 for the servers we probed (dataset $\mathcal{S}_B$). As with the router set, the general pattern is for a more open security policy in IPv6, with HTTPS, SSH, NTP, Telnet, SMB, and SNMP more open. The overall discrepancy we find in the server list between IPv4 and IPv6 is smaller than in the router data, relatively speaking. However, the sample size is twenty times larger. Thus, in absolute terms, even the smaller differences between IP version found in this dataset translate to thousands of potentially inadvertently exposed systems.

Because characteristics of the server dataset are heavily influenced by the overwhelming presence of HTTP servers, we examine the server dataset in two dimensions: all responsive servers (figure 2a) and the 191K (37%) server hosts that respond on port 80 on neither IPv4 nor IPv6 (figure 2b). Other than for NTP—which is nearly flat, going from 52% to 49% more open in IPv6, and HTTPS—which drops from 40% to 15% more open in IPv6, mostly due to the elimination of a single large hosting provider's servers—the fraction of hosts for which IPv6 is more open than IPv4 increases for every tested application in this non-HTTP subset. For instance, SSH's openness in IPv6 relative to IPv4 increases from 5% to 19% (≈4x), Telnet jumps from 46% to 112% (≈2.5x), SMB from 25% to 49% (≈2x), and SNMP from 109% to 345% (≈3x). These results suggest that dual-stacked non-web servers generally have more policy discrepancy and, thus, apparently more IPv6 vulnerability than the overall dual-stacked server population suggests. Our intuition behind examining non-HTTP-responsive servers separately stems from the fact that we believe these servers are less likely to be behind load balancers or IPv6 gateways (e.g., as offered by CloudFlare [2]). Since these load balancers and gateways generally do not forward non-web traffic to the actual server behind them and since they may terminate the IPv6 connection (in the case of gateways, that is, in fact, their function), they are much less likely to show IPv6 capability on non-web ports. Thus, looking at the non-web subset of servers may be more indicative of the *typical* configuration of the servers actually providing the content or service. As with the router list, we next discuss each application result for servers and comment on its possible impact.

HTTP and HTTPS: HTTP was less open on IPv6 than IPv4 by 7%, but there were 3.5K servers not reachable over IPv4 that were reachable on IPv6. Since it is unlikely that dual-stacked public websites would purposefully allow only access via IPv6, it is possible some of these servers are hosting non-public content. With respect to HTTPS, we did find a large percentage of servers (19%) only reachable over IPv6. Digging deeper into this peculiar group, we found that 94% of these IPv6-only HTTPS servers (92K hosts) belong to a single large European hosting provider. Of the hosts operated by this provider, 99% have HTTP open on both IPv4 and IPv6, while HTTPS is only served for IPv6. We contacted this provider but did not receive a response and, thus, have no explanation as to the intention behind this configuration.

SSH and Telnet: Both remote terminal applications were more open for IPv6, at 5% and 46% (respectively) in the overall server set, and 19% and 112% more open for IPv6 in the non-webserver set. Although the policy mismatch percentages are more modest than for routers, in absolute terms 20K servers were only reachable on SSH via IPv6 (versus only 6.5K that were reachable by IPv4). In addition, the non-webserver set shows a more worrying openness pattern, perhaps as a result of these systems having a more varied role, or our probes not being dropped by intermediate gateways. For Telnet, 2.5K were only reachable over IPv6 and 1.4K only over IPv4. This means that 23K servers could be vulnerable to brute-force password or server vulnerability exploits that were protected via IPv4. Digging a bit deeper at cross-application groups, we were curious if the IPv6-open SSH servers were more likely to also be open on Telnet, as it is used similarly to SSH and may also be neglected to be blocked by the same operators. Indeed, a disproportionate 7.3% of these SSH servers were also open to responding over IPv6 on Telnet (versus 0.5% in the overall sample that had Telnet open for only IPv6, regardless of SSH status). As SSH brute-force scanning is highly prevalent in IPv4 [30], it is reasonable to assume that such attacks over IPv6 are on the horizon. While random address scanning may not be common in IPv6 (though, see § VIII), once a hostname for a dual-stacked server is discovered, brute-force password guessing against that server over IPv6 is feasible. Since 20K servers are running SSH but have blocked it on IPv4, they may be less likely to utilize other SSH security measures.

SMB and RDP: The Server Message Block (SMB) application layer protocol is generally used by Microsoft Windows systems for file and printer sharing, as well as an inter-process communication layer. Over the years, it has been an attack vector due to numerous vulnerabilities, and has been exploited by worms, including Sasser and Conficker [40]. It is often on the Internet Storm Center's list of top-10 most scanned ports [45]. As such, this port is often treated as internal-only by operators and is commonly the subject of filtering policy [46]. The remote desktop protocol (RDP) is also built into Windows servers and clients and allows remote console access to the systems. While this application does not have as deep a history of exploits as SMB, it does provide management access to Windows systems; thus, as with SSH and Telnet, if it is exposed to connections from the public network, the potential exists for brute-force or other exploits that can lead to system compromise. In our analysis, SMB was found to be open on 25% more hosts via IPv6, exposing a total of 2.4K hosts that have it blocked on IPv4 in the overall server population. In the non-webserver population, 49% more hosts were reachable only via IPv6. RDP is less open on IPv6, but we did see nearly 700 servers with this port open for IPv6 where it was closed on IPv4.

DNS and NTP: Open DNS resolvers are problematic for two reasons. First, open resolvers can be susceptible to cache poisoning attacks [31], [48]. These, in turn, leave the users of subverted resolvers vulnerable to being re-directed to malicious services. Second, open resolvers are susceptible to being leveraged in reflection and amplification DDoS attacks (e.g., [37]). The DNS port on servers is less open via IPv6 than IPv4. A small fraction of servers, numbering 2.3K, were found reachable via only IPv6. We also found 52% more servers allowing NTP queries via IPv6 compared to IPv4. This means roughly 10K additional servers—that return system and version information—can be used as DDoS amplifiers [18] or for reconnaissance to gather version and system information

about the servers. While weaker threats, both DNS and NTP have had vulnerabilities reported as well as been used to attack others in DDoS campaigns.

FTP: We found FTP to be slightly less open (7%) in IPv6 than IPv4 in the overall server population tested, though more open (5%) in the non-webserver population. For IPv6, there were a small number (3K hosts, 0.8%) only allowing FTP connections over IPv6 (versus 3.1% only on IPv4). Interestingly, FTP's prevalence in the webserver set is more closely correlated with being on an HTTP server. For the fraction that were open in IPv6 where IPv4 was blocked, these could represent a back door to content, including source code to websites.

SNMP: Although the absolute numbers were low for SNMP among servers, we found 109% more of them (1.6K) to respond over IPv6 than IPv4; in the non-webserver population, a staggering 345% more systems were open over IPv6 where the IPv4 application was blocked. This may be a source of reconnaissance for attackers or may indicate that the default read/write *private* community is also open on these servers (which, for ethical reasons, we did not test). As such, it is concerning that an additional almost two thousand servers may be probed (and possibly manipulated) over IPv6 via SNMP.

MySQL: Finally, we probed servers for the MySQL server port, and found that only 0.5% supported IPv6 at all. MySQL prior to version 5.5.3, released in mid-2010, did not support IPv6. Current versions of the database support IPv6, but IPv6 was not enabled by default, even on dual-stacked hosts, until version 5.6.6—first released in mid-2012 [39]. In fact, when we analyzed the MySQL minor version strings returned by 32K servers that responded to our banner grab, as described in § III, 26% were running versions that did not even support IPv6, 66% were running versions with IPv6 disabled by default, and just 8% were running versions where IPv6 was supported and enabled by default. In absolute numbers, nearly 600 servers responded on IPv6 only, while 2.2K responded on both protocols and 35K responded on IPv4 only. Similar to FTP access, MySQL access is correlated with presence on a webserver, suggesting a reliance on a database system that is needlessly exposed to the Internet. In fact, since databases are typically run as back-end services to web sites or internally in organizations, the fairly high number of globally reachable servers was surprising, and the several hundred apparently reachable by IPv6 only, though relatively few, is concerning.

Overall, the server dataset showed smaller discrepancy between IPv4 and IPv6 port filtering policy for the applications we tested than we found in the router probes. However, as we noted, there were some high-value applications that were more open, and, due to the substantially larger population, the raw numbers of servers open on IPv6 only for many applications is of concern. In many cases, brute-force attacks are enabled by this discrepancy, and in other cases, known vulnerabilities in software may be exposed on thousands of dual-stack servers whose operators may believe that they have no exposure to these threats due to their IPv4 filtering.

V. Policy Uniformity

A. Network Response Uniformity

Section IV shows a difference between IPv6 policy and the *intended* policy, as indicated by the IPv4 policy. We seek to understand whether this discrepancy is a symptom of an organization's overall security posture or due to small scale misconfiguration that deviates from the organization's intended policy. Therefore, in this section we aggregate results for each organization—at both routed prefix and autonomous system granularities—and assess policy uniformity.

We aggregate hosts in our $\mathcal{R}_T$ and $\mathcal{S}_T$ datasets by routed prefix and origin AS based on BGP table data collected by RouteViews and the RIPE RIS BGP collector on February 1, 2015. We find that the IPv4 and IPv6 addresses for a host are mapped to the same AS in 94% ($\mathcal{R}_T$) and 95% ($\mathcal{S}_T$) of the cases. Therefore, for simplicity we label the hosts with their IPv4 routed prefix and AS number. Table III shows the mean and median number of devices we detect in each organization for our datasets. Further, we label each host and service with the protocol(s) that allow connection. Hosts with multiple IPs are labeled by majority. When a given service is unreachable via both versions of IP we exclude it from further analysis because we cannot determine whether the service is not reachable due to policy or simply not running, and, therefore, these cases provide no policy insight. For each service on each host we are left with one of three labels: "4" for services that are only reachable via IPv4, "6" for services that are only reachable via IPv6 and "B" for services that are reachable via both IPv4 and IPv6. Given these labels, we define the uniformity for each service within the organization—delimited by routed prefix or AS—as the fraction of hosts with the most common label for that service. For example, consider an organization with five devices running DNS, three of which are labeled "B", one labeled "4" and one labeled "6". The uniformity is therefore 60%.

TABLE III: Number of devices within an organization.

Dataset	Aggregation	Mean	Median
Router	Routed Prefixes	20	5
	Autonomous Systems	40	5
Server	Routed Prefixes	52	6
	Autonomous Systems	133	8

To put our uniformity results in perspective, we compare with a "pseudo network" which is made up of a random selection of hosts—regardless of network boundary—of the same size as the median organization size given in table III. We compute uniformity across the randomly chosen pseudo network just as we describe above. For each application, we calculate the mean uniformity across 1,000 such random pseudo networks. Figure 3 shows the mean uniformity results for both routed prefixes and random pseudo networks for both datasets. First, we find at least 90% mean consistency within organizations across applications. This indicates that the disparity we detect between IPv4 and IPv6 policy is not driven by one-off misconfigurations, but is in fact a systematic difference in policy deployment.

Additionally, the figure shows—across datasets and applications— higher uniformity within actual organizations than within randomly selected pseudo networks. This strengthens our conclusion that we are detecting in-situ policy differences and are not being lead astray by small, but broad misconfigurations. Also, we elide the results for organizations

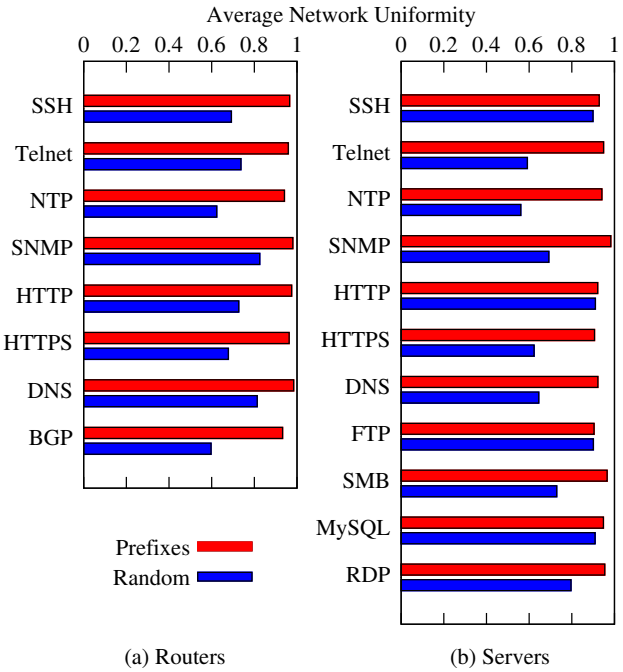


Fig. 3: Average organization uniformity for router ($\mathcal{R}_T$) and server ($\mathcal{S}_T$) dataset compared to the average pseudo-network of same median host count (each randomly selected from population of host results). Uniformity is more consistent within network boundaries than within random groupings.

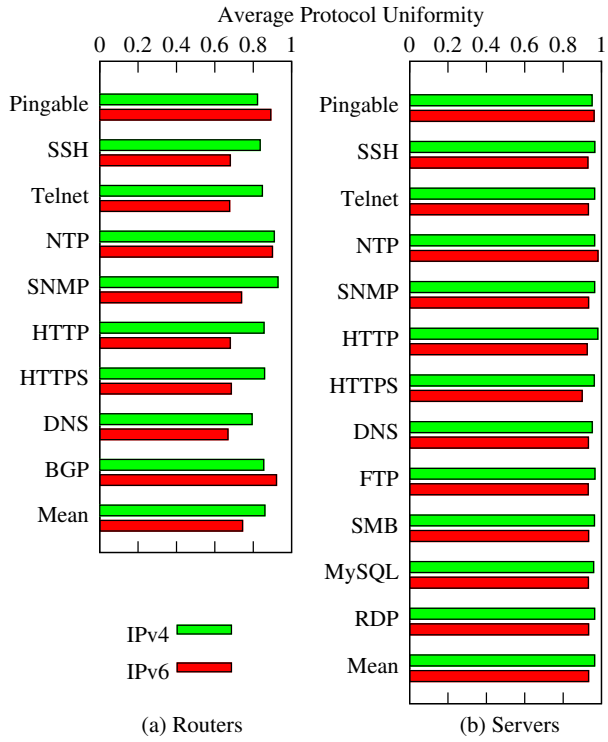


Fig. 4: Average intra-IP version uniformity within hosts having more than one IP of the given version. We see that results are more consistent for IPv4 than IPv6, and more in the server ($\mathcal{S}_T$) dataset than the router ($\mathcal{R}_T$) dataset. Also, we show that the fraction of addresses that are ICMP-pingable when multiple addresses are associated with the same host is higher for IPv6 than IPv4.

defined by AS for clarity, however note: (*i*) the uniformity is generally lower for AS-based organizations than routed prefixes due to the increased aggregation across different administrative domains, and (*ii*) just as with routed prefixes the uniformity is greater for actual organizations than for pseudo networks.

B. Intra-protocol uniformity

We next tackle an issue related to organization-level uniformity: host-level uniformity. That is, how uniform are individual hosts for the same version of IP across addresses? This question is important for two reasons. First, if policy differs for hosts across different addresses via the *same* protocol, it may not be surprising that there are differences between IPv4 and IPv6. Further, non-uniformity at the address level could indicate ad-hoc policy applied at individual machines as opposed to systematic policy at the organizational border. Second, intra-protocol uniformity speaks to the maturity of security controls and, on average, is useful in comparing the difference in maturity between protocols. For example, if we find IPv4 to be more consistent than IPv6, this may be an indication that security controls for IPv4 are more mature, tested, and robust than for IPv6.

We calculate the uniformity across each host and IP version and present the mean uniformity across hosts in Figure 4. In addition to per-application results, we also show two additional sets of bars: (*i*) the overall mean across all applications and (*ii*) the mean uniformity for ICMP ping. The plots first show that the host-level uniformity is higher for servers (90–95%) than for routers (70–90%). One possible reason for this difference is router IP addresses identify individual interfaces, which have different tasks (i.e., peering with different networks). Therefore, it may be natural to find different policy applied to different interfaces. On the other hand, servers do not have the same sort of natural per-IP division of labor and therefore show higher uniformity across addresses.

In the $\mathcal{S}_T$ dataset we find approximate parity between IPv4 and IPv6 in terms of uniformity across applications. This is in contrast to the $\mathcal{R}_T$ dataset where we generally find higher uniformity in IPv4 compared to IPv6. There are two exceptions where IPv6 is more uniform than IPv4: BGP and ICMP. While we cannot readily explain BGP's disparity, the ICMP difference may stem from ICMP being less strictly filtered in IPv6 due to a deployment requirement for IPv6 (e.g., [23]).

VI. Blocking Enforcement

Having established that myriad filtering occurs, we next turn our attention to the mechanisms employed to block traffic and where those mechanisms are implemented. To study this, we use traceroute probes, as described in § II-B, for both routers ($\mathcal{R}_T$) and servers ($\mathcal{S}_T$). For each application, and each address associated with each host, we first determine whether the application is open or closed. For each closed application we determine the enforcement mechanism. As we note in the last section, we do find cases where policy differs within the same IP protocol. In these cases, we label the host based on

the majority enforcement mechanism. We classify each attempt, as follows.

- ***Open***: In this case, the target host responds favorably (i.e., with a TCP SYN+ACK or a UDP response).
- ***Passive:Target***: In this case, the target host silently drops the SYN or UDP request. We detect this by observing that the last responding host within the traceroute is the hop immediately prior to the target host (as established by ICMP-based traceroutes and/or traceroutes involving other applications).
- ***Passive:Other***: In this case, we find that a hop in the path prior to the hop before the target host is the last hop to respond to the traceroute. Therefore, we conclude that a firewall is silently dropping the traffic before arriving at the target host.
- ***Active:Target***: In this case, the target host actively responds to our SYN or UDP request with an error indicating the service is not available (e.g., TCP reset or ICMP error message).
- ***Active:Other***: In this case, a device on the path towards the target host issues an active ICMP error or TCP reset that indicates the service is not available.

Note, firewalls typically simply drop undesired traffic silently without generating error traffic (i.e., fall in the "Passive:Other" category). Closed ports on hosts are more prone to generating an active error message (i.e., "Active:Target"). Thus, the breakdown between our various categories can shed light on firewall, access control list, or other similar policy-enforcing device in the path to the target.

TABLE IV: Connection failure mode distribution differences. We observe that connection failures are more frequently active for IPv6 than for IPv4 in both datasets, suggesting fewer silently-dropping policy devices in IPv6.

	Router ($\mathcal{R}_T$)		Server ($\mathcal{S}_T$)	
Mode	**Mean IPv4**	**Mean IPv6**	**Mean IPv4**	**Mean IPv6**
Open	4.17	6.04	18.57	18.89
Passive:Target	43.50	27.15	36.06	31.17
Passive:Other	10.12	15.82	16.31	14.20
Active:Target	30.93	36.14	22.82	27.61
Active:Other	3.55	6.94	2.09	2.79

A. Typical Connection Failure Modes

Table IV shows the average breakdown across applications into the categories above for each dataset and for IPv4 and IPv6. We first note that across dataset and IP version host-based policy enforcement accounts for the majority of the cases where traffic is filtered (i.e., the ":Target" categories). Additionally, silent dropping by the network accounts for 10–15% of the filtering cases, while active errors are relatively rare for the network to generate (2–7% of the cases). However, there are also differences between IPv4 and IPv6. For instance, IPv4 shows Passive:Target is more prevalent than Active:Target in the router dataset, but IPv6 shows the opposite. Further, an active error message is more likely in IPv6 than in IPv4—perhaps indicating traditional border firewalls are silently discarding unwanted IPv4 traffic and not yet dealing similarly with IPv6 traffic. At a high level these results show that even when both protocols implement the same high-order policy to block some service, they are not always doing so in the same manner.

While we omit individual applications' failure distributions, one interesting outlier to mention in the router dataset is NTP. We find NTP is five times more likely (24% versus 5%) to respond with an active error in IPv6 than in IPv4. Given the widespread IPv4 NTP DDoS spike and subsequent operator mitigations reported in recent years [18], we might expect silent dropping of NTP to be a prevalent security posture. However, our results suggest that while this is a reasonable expectation in IPv4, sadly this mitigation is not as common in IPv6.

B. Connection failure Locations

A second aspect of policy enforcement is the *location* of filtering. We started to address this above by detecting whether policy is applied on the host or by some on-path network element prior to the target host. In this section, we analyze cases where policy is being enforced by the network and not the host and attempt to locate where. For each failing traceroute probe not ascribed to the target, we extract the difference in the hop count between where we know that target host to be—as established via successful ICMP echo and open application responses—and the final response from the non-target. This response could be either an active error message (Active:Other) or a normal traceroute (TTL expired) probe response in the case of a silent drop (Passive:Other). Figure 5 shows the fraction of these responses at each hop distance prior to the target host.

First, we find that the differences between IPv4 and IPv6 drop distance distributions are generally small ($< 6\%$) for servers at each hop distance. Further, 49% of drops in IPv6 and 55% of drops in IPv4 happen two hops away from the server. This suggests that, when policy is applied along the path to a server and not at the server itself, it is likely to be applied at the same point for both protocols. For routers, the difference in distance distribution between IPv4 and IPv6 connection failures was greater (up to 20%). IPv6 drops are most likely to happen at a distance of three hops away. Conversely, IPv4 is most likely to see drops two hops from the target. The distribution at earlier and later hops shows rough parity between IPv4 and IPv6. In sum, although the differences between IPv4 and IPv6 enforcement location are not stark in general, we did find some differences which, when combined with the connection failure mode distributions we show above, lead us to conclude that deployment of policy enforcement mechanisms, both in number, kind, and to a lesser extent, location, differs measurably between the two protocols.

VII. Validation and Case Studies

We solicited validation on our methods and our findings from 16 networks for which we had contacts. These networks were varied in their types, including access, transit, university, content, and hosting networks, and they varied in their location footprints, ranging from Asia, Europe, Oceania, and North America. For each network, we emailed our findings with a project summary, listing IPv4/IPv6/name tuples with associated information on which ports were apparently blocked in IPv4 but were not blocked in IPv6. We received responses from twelve

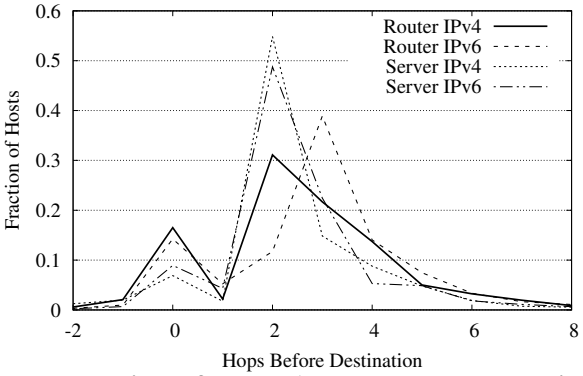


Fig. 5: Fraction of hosts (mean across all applications in dataset) where failure response (*Passive:Other* or *Active:Other*) originated given hops prior to target.

TABLE V: Validation summary. Twelve of sixteen operators of various types responded, and each indicated that discrepancy was unintentional. Ten took steps to remediate.

Operator	Host-App Pairs w/Only IPv6 Open	Response
Global CDN 1	3	✓
Tier1 ISP 1	498	
Global Transit Pro. 1	201	✓
Large Hosting Pro. 1	<800	
Large University 1	5	✓
Large University 2	6	✓
Large University 3	989	✓
National ISP 1	4757	✓
National ISP 2	89	
Research/Ed. ISP 1	1	✓
Research/Ed. ISP 2	523	✓
Research/Ed. ISP 3	77	✓
Research/Ed. ISP 4	17	✓
Small Hosting Pro. 1	17	✓
Small ISP 1	12	
Small Transit Pro. 1	2	✓

networks, summarized in Table V. In every case, we received a confirmation of our hypothesis that the underlying cause was an oversight on consistent application of security policy. In addition, ten of the twelve responding networks immediately worked to establish a congruent policy in IPv6.

When we followed up with individual operators, we found that policy was typically being applied on the individual devices. One operator had used IPv4-specific examples for how to harden the control plane of a router, without adding additional configuration to accomplish the same in IPv6. Another operator had an organization-wide standard security policy for IPv6 that was found to not be applied to a single device; this device was installed as a IPv4-only system, and had IPv6 later added. The organization had been working to ensure their IPv6 posture was on par with their IPv4 posture, though the firewall configuration tool their system administrators had been using does not have an IPv6 option, leaving a lot more manual work for them. Similarly, a large transit provider confirmed that they did not intend for external SSH and telnet access for their routers in IPv6. They deployed configuration on the routers to prevent external access in IPv6, but were not able to deploy the same configuration on customer routers that used their address space to number the interconnection links with their customers. We also found most transit providers do not block packets headed towards their customer's interfaces.

Before this paper was published, we took additional steps to send emails to abuse contacts for 396 remaining autonomous systems not associated with the 16 operators above, whose routers were also found with open IPv6 access in $\mathcal{R}_T$, as we deemed the threat to routers to be of greatest urgency.

VIII. Scanning Feasibility

As we have shown, IPv6 often provides access to application ports that are unreachable via IPv4. This in turn provides attackers with a path to vulnerable services. However, an attacker must first find these hosts and services before attempting to exploit vulnerabilities. Within IPv4, the most straightforward method is to scan the entire address space for vulnerable services. Current scanning techniques allow a single host connected to a fast network to scan the entire IPv4 address space in less than one hour [22]. Scanning the IPv6 address space in this fashion would take on the order of 10^{22} years. The task is prohibitively expensive, even considering parallelizing the work and assuming massive network capacity improvements. Alternatively, an attacker could leverage the sorts of DNS and traceroute data we start with to form a hit list for scanning. Although this is useful to obtain a sample that is suitable for understanding the general policy posture of the IPv6 network, it is far from comprehensive.

Although scanning each IPv6 address is impracticable, some researchers note that feasibility of targeted IPv6 scanning depends on the device addressing strategy within each block [16]. When operators concentrate devices in a contiguous sub-block of a routed prefix, attackers can concentrate on the sub-block and ignore everything outside—potentially putting the task of comprehensive scanning of devices within reach. Random address assignment within a routed prefix may at first appear as security-through-obscurity, but the strategy actually determines whether IPv6 brute-force scanning is practically possible. As an example, 2008 work by Malone showed that significant fractions of the host ID portion of IPv6 addresses were derived from the MAC address using the EUI-64 mechanism [38]. This addressing strategy effectively reduces the search space for an attacker from 64 bits to 48 bits—and even further down to 24 bits if the Ethernet card vendor is known or can be guessed. Further, Malone notes a prevalence of low-integer host IDs. While EUI-64 is less common in today's networks, we are still interested in whether current address assignment strategies impact an organization's security posture.

Therefore, we next turn to using addresses found in our $\mathcal{R}_T$ and $\mathcal{S}_T$ datasets to understand the addressing practices of operators. We first note that the high-order network ID portion of routed IPv6 networks is advertised in BGP—often with a prefix of /48 or longer. These are available in public routing table repositories and serve to significantly winnow the scanning space an attacker would have to cover for a comprehensive scan. After the prefix, the natural next question is whether the middle (subnet) portion of the IPv6 address—typically 16 bits—is random. In our target data, we find that 47% of the router and 45% of the server subnets use only the lower 8 bits. Additionally, 8% of router subnets and 19% for servers use a reverse-low pattern, where the high-order four bits are used and the remainder of the bits are zero, resulting in 15 possible subnets. Thus, just 270 possibilities account for 55% of router and 64% of server subnets. Scanning this small fraction (0.4%)

of the theoretical 65,536 possible subnets would have identified the majority of used networks in our DNS-derived sample.

TABLE VI: IPv6 Interface Identifier (IID) types for all IPv6 addresses for hosts in the $\mathcal{R}_T$ and $\mathcal{S}_T$ datasets, including 30K router and 968K server IPv6 addreses. For each dataset, we show the percentage in that category and a cumulative total. We find that 89% of router and 37% of server addresses are within very low ranges, allowing discovery within seconds on a subnet. Recall that half of the bit space (i.e., 32 bits) is a minute fraction of the IPv6 address space—$2^{32}/2^{64}$ or 1/4,294,967,296

		Router		Server	
IID Bits Used	IID Value Range	%	Cum. %	%	Cum. %
1	<= 0x0001	23.74	23.74	5.83	5.83
4	<= 0x000F	37.89	61.63	5.94	11.77
8	<= 0x00FF	6.87	68.49	4.76	16.53
16	<= 0xFFFF	11.00	79.50	5.50	22.03
32	<= 0xFFFF FFFF	9.81	89.31	14.50	36.53
EUI-64	Middle == 0xFFFE	0.92	90.23	4.92	41.45
Other	Not in Above	9.77	100.00	58.55	100.00

Finally, we turn to the host ID portion—low-order 64 bits—of each address. In table VI we classify each address into one of several allocation ranges based on use of a decreasing number of leading zeros or use of the EUI-64 scheme. We find nearly a quarter of routers and 6% of servers use the value of 1 as the host id, and that scanning just the lower quarter—16 bits—of the theoretical host ID space will identify 80% of the open routers and 22% of the open servers. The address assignments are therefore extremely concentrated and an attacker could get significant coverage at a miniscule fraction of the cost of a full scan ($2^{16}/2^{64}$ or just 4×10^{-15} of addresses). Further, we find EUI-64-derived addresses in just under 1% of the routers and nearly 5% of servers. We also find that the 24-bit vendor ID portion of the host ID shows eight vendors account for 46% of the routers and 69% of the servers. This reduces the search space to the low-order 24 bits, which, even with random assignment, is tractable to scan.[2] A 1 Gbps (1.4 Mpps) scanner [22], could scan all of the categories in table VI except for "Other" and including only the top eight most common EUI-64 vendor IDs on any given subnet in 53 minutes. In our dataset this would identify 90% of routers and 40% of servers at a minuscule fraction of the cost of scanning a full IPv6 64-bit address block at that rate (418 thousand years). Given these numbers and the addressing schemes we saw, brute force scanning for servers and routers, while not exhaustive or foolproof, is still largely feasible for enumerating the majority of IPv6 hosts on a subnet. With prefixes easily identifiable and most subnets using just one of 270 values, we conclude that scanning is still a viable way to identify large fractions of hosts within networks, even if complete scanning of the IPv6 address space is impracticable. Thus, our main findings reporting greater openness in IPv6 may be exploitable not just by hostname but also via brute force scans, especially if they target a single network prefix. One word of caution for researchers interested in applying scanning as a technique for brute-force measurement in IPv6: there is a known severe denial of service condition that can be triggered in many older or improperly configured IPv6 routers due to memory exhaustion from incomplete neighbor discovery entries (see e.g., [5]).

[2]MAC addresses are often assigned sequentially as the network cards are manufactured. Thus, they are not uniformly random, and one can expect to find less entropy within large organizations in EUI-64-derived IP addresses [28].

IX. Related Work

Standards and deployment guides (e.g., [14], [23], [26], [32]) have been urging operators to apply firewall rules and access control lists for IPv6 in parity with IPv4 as part of their deployment of IPv6. Unfortunately, security researchers as well as RFC authors have lamented that in practice: "networks tend to overlook IPv6 security controls: [often] there is no parity in the security controls [between] IPv6 and IPv4" [5], and "in new IPv6 deployments it has been common to see IPv6 traffic enabled but none of the typical access control mechanisms enabled for IPv6" [15]. Beyond the quotes, we were not aware of any data that would shed light on the extent of real-world deployment of security filtering for IPv6. A desire to measure and raise awareness about these fundamental security control disparities in IPv6 was the motivation for our work. To our knowledge, ours is the first such Internet-scale study of deployed IPv6 security policies.

The IPv6 protocol was standardized nearly twenty years ago, before many lessons had been learned about Internet security. On top of that, IPv6 introduces many changes and features that go far beyond increased address space. As such, issues with the design and implementation of IPv6 have come to the fore as IPv6 is given more scrutiny by early adopters and researchers. For instance, Ullrich *et al.*, aggregated a taxonomy of 36 known design and implementation weaknesses in IPv6 [51]. Several of these problems (e.g., fragmentation header related, hop-by-hop header) have been discussed for some time by practitioners and non-academic security researchers (e.g., [8]), and there have been a number of updates to the original IPv6 specifications in recent years as a result (e.g., [7], [25], [27]). However, hardware changes to support the updated specifications in policy enforcement devices are taking time for vendors to implement and operators to deploy (e.g., [5]), leaving some networks vulnerable. A carefully controlled study of such vulnerabilities would be interesting future work. Our paper, however, focused on characterizing the apparent misalignment of network security policy between IPv4 and IPv6, as measured by relatively reachable application ports. This shines light on the vulnerability that IPv6 poses as a *path* to exploit *upper layer applications*, rather than exploring weaknesses in IPv6 itself. As such, our work is largely orthogonal to such research and standards changes.

There has been significant interest in characterizing the size of the open or vulnerable IPv4 host population for various ports, with several recent studies related to the topic of large-scale IPv4 application discovery (e.g., [21], [44]). Our study seeks an at-scale measurement of commonly vulnerable or high-value open applications on IPv6 versus IPv4. As recent studies (e.g., [17], [20]) and data (e.g., [29]) have shown a surge in IPv6 deployment, security weaknesses related to the rise of IPv6 are naturally of interest to the network security community. Ours is the first large-scale study of the degree to which dual-stacked hosts provide the same services across both protocols, a metric pertinent to the study of IPv6 adoption.

IPv6 host addressing schemes deployed in the wild were last studied at scale by Malone in 2008 [38], though much has changed since that study, including several orders of magnitude more IPv6 deployment and the phasing out of EUI-64-based host identifiers by common operating systems. However, the results from Malone's September 2007 traceroute data, which is likely dominated by routers, shows 80-90% of host IDs there using just the lower 8 bits, suggesting some improvement in desirable randomness of HIDs in the intervening eight years (we saw 68%). Methods for discovering IPv6 hosts for scanning via leveraging secondary information sources such as the DNS (earlier discussed by Bellovin *et al.* [11], then in RFC 5157 [16], and more recently in an IETF draft [28]) have been successfully applied to IPv6 client discovery in recent years (e.g., [41]). Our analysis of the host IDs used by servers and routers in particular, showed secondary sources were not necessary for identifying large fractions of these high value hosts given today's address allocation patterns. This should help dispel the myth that simple scanning on IPv6 is futile, and it somewhat heightens the risk associated with our main application openness findings.

X. Discussion

We built lists of 25K dual-stacked pingable routers and 520K dual-stacked pingable servers and tested connectivity over IPv4 and IPv6 to a set of common application ports. Our experiments showed that for both routers and servers, 26% of the hosts were more open for IPv6 than for IPv4 for at least one tested application (versus 18% and 17%, respectively, that were more open for IPv4). For routers, the average application was open for 84% more hosts via IPv6 than IPv4, including SSH, which was reachable via IPv6 for a staggering 166% more routers than over IPv4. For servers, this number was a lower but still significant 12%, which notably included SSH (5%), and Telnet (46%). The numbers were even higher for the 37% of servers that did not support HTTP (and, thus, were less likely to be behind load balancers or gateways). Among those 191K servers, 49% more servers were open for IPv6 than IPv4 on the often-attacked server message block (SMB) protocol, 112% for Telnet, and 343% for SNMP, for example. Lastly, deeper probing we conducted using traceroutes also showed that even when both protocols blocked an application, the manner in which policy is deployed (i.e., discrete firewall or host firewall) also differs between IPv4 and IPv6.

Even when IPv6 was less open, there were hundreds or thousands of hosts for many applications that were only reachable via IPv6. While we can speculate that hosts which only support a service on IPv4 have yet to configure IPv6, it is more difficult to imagine plausible scenarios where a service is not intended to be available on IPv4 but is intentionally so on IPv6. This is the reason we are concerned even when, for applications where there are relatively more IPv4-reachable hosts, we still find hundreds or thousands only accessible over IPv6. While the lack of IPv6 connectivity may be an adoption problem, it is not a security problem; whereas, each of the hosts that do have a service reachable over IPv6 only, even if they are a minority of the hosts, could be exposing the system to an unexpected attack vector—a back door waiting for an IPv6-savvy attacker to come along and knock on it.

To have more confidence in both our findings and our assumptions of policy parity intent, we contacted a sample of network operators who had applications reachable over IPv6 but not IPv4. We received responses from twelve of sixteen networks contacted. They validated that (*i*) our mappings between IPv4 and IPv6 generally appeared correct, that (*ii*) indeed these applications were reachable, and that (*iii*) the openness on IPv6 was, in fact, not intentional.

We note that the risk due to these services being reachable—where intended policy appears to be that they are not—is likely exacerbated by the lack of maturity of IPv6 tools and processes. For instance, older Netflow version 5 systems, which are essential elements of aggregating, transmitting, and storing network traffic data for many network operators, do not support IPv6 (the newer Cisco Netflow v9 and IETF standard IPFIX do), requiring both sources of flow information and sinks to be updated to have visibility into IPv6 traffic. Aside from Netflow, anecdotal evidence suggests some large organizations, including service providers, run various homegrown or legacy network management software that simply does not yet support IPv6.

There is growing awareness of the fundamental yet basic challenges in securely operating IPv6 networks—including, for example, due to address presentation differences and IPv6 address agility:

> In an IPv4 network, it is easy to correlate multiple logs, for example to find events related to a specific IPv4 address. A simple Unix grep command was enough to scan through multiple text-based files and extract all lines relevant to a specific IPv4 address.
> In an IPv6 network, this is slightly more difficult because different character strings can express the same IPv6 address. Therefore, the simple Unix grep command cannot be used. Moreover, an IPv6 node can have multiple IPv6 addresses...
> In order to do correlation in IPv6-related logs, it is advised to have all logs with canonical IPv6 addresses. Then, the neighbor cache current (or historical) data set must be searched to find the data-link layer address of the IPv6 address. Then, the current and historical neighbor cache data sets must be searched for all IPv6 addresses associated to this data-link layer address: this is the search set. The last step is to search in all log files (containing only IPv6 address in canonical format) for any IPv6 addresses in the search set. [15]

We highlight this quote to underline that the deployment of IPv6 has implications far exceeding merely a *larger* address space. The complexity introduced by its many features as well as lack of stack robustness, process maturity, and available tooling that come from decades of deployment for IPv4 are evident both in the number of issues and standards changes seen in recent years and the breadth of deployments we were able to find with IPv6 unprotected. We call on operators and researchers to give IPv6 security a deeper look.

Acknowledgments

This work was supported in part by the National Science Foundation under contracts CNS 1111449, CNS 1111672, CNS 1111699, CNS 1213157, CNS 1237265, CNS 1505790, and CNS 1518741, and by Intel Corporation. We would like to thank Young Hyun at CAIDA for helping run DNS lookups, and HD Moore, Mark Schloesser, and Zakir Durumeric for

DNS data. We are also grateful to the network operators that responded to our validation emails. Matthew Luckie conducted part of this work while at CAIDA, UC San Diego.

REFERENCES

[1] Cisco Nexus 7000 Series NX-OS System Management Configuration Guide, Release 5.x. http://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/5_x/nx-os/system_management/configuration/guide/sm_nx_os_cg/sm_3ntp.html#wp1107779.

[2] CloudFlare : IPv6 Gateway Feature. https://www.cloudflare.com/ipv6.

[3] NMAP : ssl-enum-ciphers. https://nmap.org//nsedoc/scripts/ssl-enum-ciphers.html.

[4] Scamper. http://www.caida.org/tools/measurement/scamper/.

[5] Security Assessments of IPv6 Networks and Firewalls. Frankfurt, Germany, June 2013. Slides of Presentation at IPv6 Kongress.

[6] World IPv6 Launch : Network Operator Measurements July 8, 2015. http://www.worldipv6launch.org/measurements, 2015.

[7] J. Abley, P. Savola, and G. Neville-Neil. Deprecation of Type 0 Routing Headers in IPv6. RFC 5095, 2007.

[8] A. Atlasis. Attacking IPv6 Implementation Using Fragmentation. https://media.blackhat.com/ad-12/Atlasis/bh-ad-12-security-impacts-atlasis-wp.pdf, 2012.

[9] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. The Menlo report. *IEEE Security & Privacy*, 10(2):71–75, 2012.

[10] D. Barr. Common DNS Operational and Configuration Errors. RFC 1912, 1996.

[11] S. M. Bellovin, B. Cheswick, and A. Keromytis. Worm propagation strategies in an IPv6 Internet. *LOGIN: The USENIX Magazine*, 31(1):70–76, 2006.

[12] R. Beverly and A. Berger. Server Siblings: Identifying Shared IPv4/IPv6 Infrastructure via Active Fingerprinting. In *Proceedings of the Sixteenth Passive and Active Measurement Conference*, PAM'15, 2015.

[13] CAIDA. Archipelago (Ark) Measurement Infrastructure. http://www.caida.org/projects/ark/.

[14] K. Chittimaneni, T. Chown, L. Howard, V. Kuarsingh, Y. Pouffary, and E. Vyncke. RFC 7381: Enterprise IPv6 Deployment Guidelines, 2014.

[15] K. Chittimaneni, M. Kaeo, and E. Vyncke. Operational Security Considerations for IPv6 Networks. 2015.

[16] T. Chown. IPv6 Implications for Network Scanning. RFC 5157, 2008.

[17] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey. Measuring IPv6 Adoption. In *Proceedings of the 2014 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM'14, 2014.

[18] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, IMC'14, 2014.

[19] E. Davies, S. Krishnan, and P. Savola. IPv6 Transition/Coexistence Security Considerations. RFC 4942, 2007.

[20] A. Dhamdhere, M. Luckie, B. Huffaker, k claffy, A. Elmokashfi, and E. Aben. Measuring the deployment of IPv6: Topology, routing and performance. In *Proceedings of the 12th ACM SIGCOMM conference on Internet measurement*, IMC'12, 2012.

[21] Z. Durumeric, M. Bailey, and J. A. Halderman. An Internet-wide view of Internet-wide scanning. In *Proceedings of the USENIX Security Symposium*, SEC'14, 2014.

[22] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *Presented as part of the 22nd USENIX Security Symposium*, Washington, D.C., 2013.

[23] S. Frankel, R. Graveman, J. Pearce, and M. Rooks. Guidelines for the secure deployment of IPv6. *NIST Special Publication*, 800-119, 2010.

[24] S. Gallagher. NSA hacker in residence dishes on how to hunt system admins, Mar 2014. http://arstechnica.com/security/2014/03/nsa-hacker-in-residence-dishes-on-how-to-hunt-system-admins/.

[25] F. Gont. Processing of IPv6 "Atomic" Fragments. RFC 6946, 2013.

[26] F. Gont and W. Liu. Security Implications of IPv6 on IPv4 Networks. RFC 7123, 2014.

[27] F. Gont, V. Manral, and R. Bonica. Implications of Oversized IPv6 Header Chains. RFC 7112, 2014.

[28] F. Gont and T.Chown. Network Reconnaissance in IPv6 Networks. Internet-Draft draft-ietf-opsec-ipv6-host-scanning-07, 2015.

[29] Google. IPv6 Statistics. http://www.google.com/intl/en/ipv6/statistics.

[30] M. Javed and V. Paxson. Detecting stealthy, distributed ssh brute-forcing. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, SIGSAC'13, 2013.

[31] D. Kaminsky. Black Ops 2008: It's the End of the Cache As We Know It. *Black Hat USA*, 2008.

[32] C. M. Keliiaa and V. N. McLane. Cyberspace Modernization: An Internet Protocol Planning Advisory. *SANDIA Report*, SAND2014-5032, 2014.

[33] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: illuminating the edge network. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, IMC'10, 2010.

[34] G. Lindberg. Anti-Spam Recommendations for SMTP MTAs. RFC 2505, 1999.

[35] M. Luckie. Scamper: a scalable and extensible packet prober for active measurement of the Internet. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC'10, 2010.

[36] M. Luckie, R. Beverly, T. Wu, M. Allman, and k claffy. Resilience of deployed TCP to blind attacks. In *Proceedings of the 15th ACM SIGCOMM conference on Internet measurement*, IMC'15, 2015.

[37] D. MacFarland, C. Shue, and A. Kalafut. Characterizing Optimal DNS Amplification Attacks and Effective Mitigation. In *Passive and Active Measurement Conference*, Mar. 2015.

[38] D. Malone. Observations of IPv6 addresses. In *Passive and Active Network Measurement*, PAM'08, pages 21–30. Springer, 2008.

[39] MySQL IPv6 Support. https://dev.mysql.com/doc/refman/5.5/en/ipv6-server-config.html.

[40] P. Porras, H. Saidi, and V. Yegneswaran. An Analysis of Conficker's Logic and Rendezvous Points. Technical report, SRI International, 2009.

[41] H. Rafiee, C. Mueller, L. Niemeier, J. Streek, C. Sterz, and C. Meinel. A Flexible Framework for Detecting IPv6 Vulnerabilities. In *Proceedings of the 6th International Conference on Security of Information and Networks*, SIN '13, 2013.

[42] Rapid7. DNS consumer hostname filtering code, 2015. https://github.com/rapid7/dap/blob/master/lib/dap/filter/names.rb#L98.

[43] RIPE NCC. Routing Information Service (RIS). http://www.ripe.net/ris/.

[44] C. Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'14, 2014.

[45] Top Ten Reports. https://isc.sans.edu//top10.html, 2015.

[46] M. Sargent, J. Czyz, M. Allman, and M. Bailey. On The Power and Limitations of Detecting Network Filtering via Passive Observation. In *Proceedings of the Passive and Active Measurement Conference*, PAM'15, 2015.

[47] Scans.io: Rapid7. DNS Records (ANY) Datasets, 2015. https://scans.io/study/sonar.fdns.

[48] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. Assessing DNS Vulnerability to Record Injection. In *Passive and Active Measurement Conference*, Mar. 2014.

[49] O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 2011.

[50] The Spamhaus Project - PBL. http://www.spamhaus.org/pbl/.

[51] J. Ullrich, K. Krombholz, H. Hobel, A. Dabrowski, and E. Weippl. IPv6 Security: Attacks and Countermeasures in a Nutshell. In *Proceedings of the USENIX Workshop on Offensive Technologies*, WOOT'14, 2014.

[52] University of Oregon. Route Views project. http://www.routeviews.org/.