# Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists

Oliver Gasser[1], Quirin Scheitle[1], Pawel Foremski[2], Qasim Lone[3],
Maciej Korczynski[4], Stephen D. Strowes[5], Luuk Hendriks[6], Georg Carle[1]
[1] TU Munich, [2] IITiS PAN, [3] TU Delft, [4] Univ. Grenoble Alpes, [5] RIPE NCC, [6] University of Twente

## ABSTRACT

Network measurements are an important tool in understanding the Internet. Due to the expanse of the IPv6 address space, exhaustive scans as in IPv4 are not possible for IPv6. In recent years, several studies proposed to use target lists of IPv6 addresses, called hitlists.

In this paper, we show that addresses in IPv6 hitlists are heavily clustered. We present novel techniques that allow to push IPv6 hitlists from quantity to quality. We perform a longitudinal active measurement study over 6 months, targeting more than 50 M addresses. We develop a rigorous method to detect aliased prefixes, which identifies 1.5 % of our prefixes as aliased, pertaining to about half of our target addresses. Using entropy clustering, we group the entire hitlist into just 6 distinct addressing schemes. Furthermore, we perform client measurements by leveraging crowdsourcing.

To encourage reproducibility in network measurement research and serve as a starting point for future IPv6 studies, we publish source code, analysis tools, and data.

## 1. INTRODUCTION

Internet scanning has a rich history of generating insights for security, topology, routing, and many other fields. Advances in software and link speeds in recent years permit to easily scan the entire IPv4 Internet in a few minutes [2, 18, 44]. However, scanning the expanse of the entire IPv6 Internet is infeasible due to its size, which is magnitudes above of what can possibly be sent or stored. Hence, state-of-the-art IPv6 Internet scanning resorts to the methods used in the early days of IPv4 Internet scanning, *i.e.*, using lists of target IP addresses, so-called hitlists [13, 15, 20].

The IPv6 address space also comes with unique, different challenges to such hitlists. First, addresses may be used only for very brief periods of time, as there is no pressure for re-use. Second, due to likewise large allocation sizes, a single network—or even a single machine[1]—can easily overwhelm a hitlist with uncountable numbers of IP addresses. Hence, a key quality of IPv6 hitlists is not the count of IP addresses, but that the hitlist is well balanced across Autonomous Systems and Prefixes, and that IP addresses in it are responsive. In this paper, we systematically tackle these challenges by:

**Comprehensive Address Discovery:** The first step in unbiasing a hitlist is creating a *comprehensive* hitlist, for which we draw IP addresses from a multitude of state-of-the-art sources, cf. Section 3.

**Clustering by Entropy:** To discover and understand clusters in the expanse of the IPv6 space, we leverage entropy analysis of IPv6 addresses. This helps to determine addressing schemes and aggregate clusters, which we explore in Section 4.

**De-Aliasing:** To reduce the potentially thwarting impact of aliased prefixes—*i.e.*, a single machine responding to all addresses in a possibly large prefix—we postulate and implement a rigorous method for Aliased Prefix Detection, which we present in Section 5.

**Longitudinal Stability Probing:** To find reliably responsive addresses, we conduct longitudinal scans for our hitlist across several protocols. As expected, we find only a fraction of discovered IP addresses to actually respond to probing, which is an important filtering criterion for curation of an unbiased hitlist (cf. Section 6).

**Going Beyond:** We also evaluate three orthogonal methods to push the frontier of IPv6 hitlists: generating addresses using Entropy/IP [24] and 6Gen [37] (in Section 7), leveraging reverse DNS records [22] (in Section 8), and crowdsourcing client IP addresses (in Section 9).

**Plotting for Exploratory Analysis:** Visualizing IPv6 datasets is challenging, as IPv4 approaches do not scale for IPv6, e.g., describing the entire address space with a Hilbert Curve. We apply the concept of squarified tree-maps [12], extended with recursive properties, to develop a space-filling plotting technique that works with selective input instead of the entire address space. This visualization technique aids specifically in exploratory analysis of IPv6 datasets, and we use it throughout the paper to give an intuitive view on our data.

**Publication and Sharing:** Curation and open sharing of a reliable IPv6 hitlist supported various scientific studies [3, 49]. We publicly share daily snapshots of our curated and unbiased IPv6 hitlist.

[1] We have indeed seen a web server responding to an entire /32 prefix, *i.e.*, $2^{96}$ addresses.

Throughout our work, we adhere to highest ethical standards (cf. Section 10.1), and aim for our work to be fully reproducible. We share code and data under:

https://ipv6hitlist.github.io/

## 2. PREVIOUS WORK

The IPv4 space is dense, with over 900 M address responsive according to public data [14] as of April 2018. The IPv6 space, however, is extremely sparse. Here, we survey previous work on targeting IPv6 and compare our work with the state of the art in Table 1.

**DNS Techniques:** The DNS was long known to be a possible source for IPv6 addresses [27,55]. Strowes proposes to use the *in-addr.arpa* IPv4 rDNS tree to gather names that may be resolved to IPv6 addresses [52], on an assumption that naming is common between protocols. This discovered 965 k IPv6 addresses located in some 5,531 ASes, many of which (56.7%) were responsive. More recently, Fiebig *et al.* walked the rDNS tree to obtain 2.8 M IPv6 addresses [22,23]. They did not, however, probe these addresses, leaving the question of responsiveness open. Borgolte *et al.* go one step further by using NSEC walking in DNSSEC signed zones reverse zones to find IPv6 addresses [11].

In this work, we evaluate the responsiveness of IPv6 rDNS as a source and find that rDNS IPv6 addresses are a valuable addition to a hitlist.

**Structural Properties:** The IPv6 address space may be sparsely populated, but address plans inside networks tend to indicate structure. Recent related work leverages the structure of IPv6 addressing schemes to find new addresses. Ullrich *et al.* use rule mining to find a few hundred IPv6 addresses [54]. With Entropy/IP, Foremski *et al.* present a machine learning approach which trains on collected IPv6 addresses to build an addressing scheme model and generate new addresses [24]. In this work, we refine Entropy/IP to generate IPv6 addresses for probing, and we introduce a new entropy clustering technique. In 2017, Murdock *et al.* presented 6Gen to find dense regions in the IPv6 address space and generate neighboring addresses [37]. In this work, we use 6Gen to generate addresses based on our hitlist and compare its performance with Entropy/IP. Murdock *et al.* also performed a basic variant of aliased prefix detection (APD), which we extend in this work. Plonka and Berger harness the structure from IPv6 address plans as a means to allow large datasets to be shared [40].

Table 1: Comparing this work with four previous works (ordered chronologically) based on the following metrics: number of addresses from public sources (#publ.), BGP prefixes (#pfx.), ASes; addresses from private sources (#priv.); include client addresses (Cts), perform active probing (Prob.), perform aliased prefix detection (APD).

| Previous work | #publ. | #pfx. | #ASes | #priv. | Cts | Prob. | APD |
|---|---|---|---|---|---|---|---|
| Gasser *et al.* [26] | 2.7 M | 5.8 k | 8.6 k | 149 M | ✓ | ✓ | ✗ |
| Foremski *et al.* [24] | 620 k | <100[1] | <100[1] | 3.5 G | ✓ | ✓ | ✗ |
| Fiebig *et al.* [22] | 2.8 M | n/a[2] | n/a | 0 | ✓ | ✗ | ✗ |
| Murdock *et al.* [37][3] | 1.0 M | 2.8 k | 2.4 k | 0 | ✓ | ✓ | ○ |
| **This work** | 55.1 M | 25.5 k | 10.9 k | 0 | ✓ | ✓ | ✓ |

1: 15 networks, with few prefixes and ASes. 2: 582 k /64s. 3: Responsive addresses.

**Hitlists:** Gasser *et al.* [26] assemble a hitlist from a multitude of sources. The vast majority of their 149 M IPv6 addresses, however, is obtained from non-public passive sources. We build upon their approach, but exclusively use publicly available sources in order to make our work reproducible.

**Crowdsourcing:** There are a few studies that have leveraged crowdsourcing platforms to perform network measurements [29,31,56]. We build on prior work by Huz *et al.* [29], who used the Amazon Mechanical Turk platform to test broadband speeds and IPv6 adoption. Compared to Huz *et al.*'s 38 IPv6 addresses, we find magnitudes more.

## 3. IPV6 HITLIST SOURCES

We leverage a variety of sources, for which we provide an overview in Table 2. Our guiding principle in selecting sources was that they should be *public*, *i.e.*, accessible to anyone for free. We consider data with an open and usually positive access decision process as public, such as, e.g., Verisign's process to access zone files. We also aim to have balanced sources, which include servers, routers, and a share of clients.

Figure 1a shows the cumulative runup of sources over time. First, we can see a strong growth of IPv6 addresses in all sources: typically an increase in factors of 10–100 over the course of a year. Second, DNS-based sources—likely revealing server addresses—together with scamper addresses clearly dominate the overall dataset.

The sources we leverage are as follows:

**Domain Lists:** Described in [4,25,46], with a total of 212M domains from various large zones, resolved for AAAA records on a daily basis, yielding about 9.8 M unique source IP addresses. This source also includes domains extracted from blacklists provided by Spamhaus [51], APWG [6], and Phishtank [39], which leverage 8.5 M, 376 k, and 170 k domains, respectively.

**FDNS:** A comprehensive set of AAAA lookups performed by Rapid7 [42], yielding 2.5 M unique IPs.

**CT:** DNS domains extracted from SSL certificates logged in Certificate Transparency, and not already part of domain lists, which yields another 16.2 M addresses.

**AXFR and TLDR:** The set of IPv6 addresses obtained from DNS AXFR transfers both from the TLDR project [34] and by running our own AXFR transfers. Obtained domain names are also resolved for AAAA records daily, which yields 0.5 M unique IPv6 addresses.

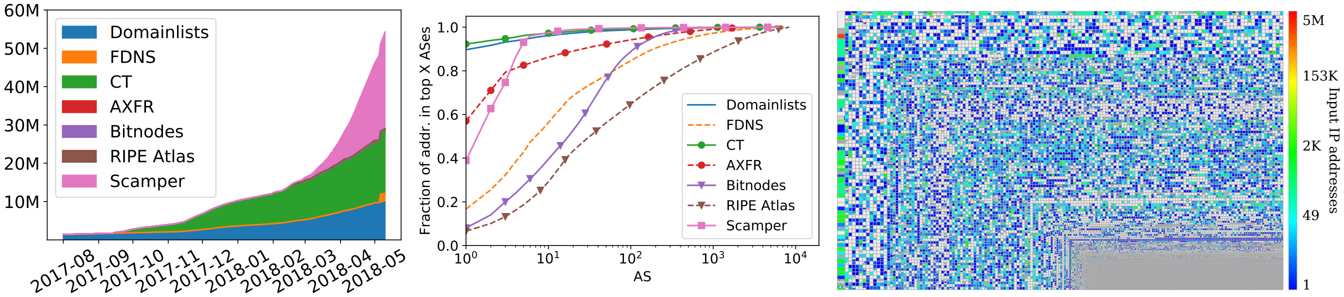
**Bitnodes:** To gather client IPv6 addresses, we use

Table 2: Overview of hitlist sources, as of 2018-05-11.

| Name | Public | Nature | IPs | new IPs | #ASes | #PFXes | Top AS1 | Top AS2 | Top AS3 |
|---|---|---|---|---|---|---|---|---|---|
| DL: Domain Lists[1] | Yes | Servers | 9.8 M | 9.8 M | 6.1 k | 10.3 k | 89.7 %★ | 2.0 %● | 1.5 %■ |
| FDNS: Rapid7 FDNS | Yes | Servers | 3.3 M | 2.5 M | 7.7 k | 13.6 k | 16.7 %★ | 8.9 %▲ | 6.7 %✣ |
| CT: Domains from CT logs[2] | Yes | Servers | 18.5 M | 16.2 M | 5.3 k | 8.7 k | 92.3 %★ | 1.6 %✣ | 0.8 %★ |
| AXFR: AXFR&TLDR | Yes | Mixed | 0.7 M | 0.5 M | 3.2 k | 4.7 k | 57.0 %★ | 14.0 %● | 8.3 %■ |
| BIT: Bitnodes | Yes | Mixed | 31 k | 27 k | 695 | 1.4 k | 8.0 %★ | 6.0 %■ | 6.0 %▲ |
| RA: RIPE Atlas[3] | Yes | Routers | 0.2 M | 0.2 M | 8.4 k | 19.1 k | 6.6 %✣ | 3.5 %★ | 3.1 %✣ |
| Scamper | – | Routers | 26.0 M | 25.9 M | 6.3 k | 9.8 k | 38.9 %★ | 23.8 %● | 12.0 %■ |
| Total | | | 58.5 M | 55.1 M | 10.9 k | 25.5 k | 45.4 %★ | 18.4 %★ | 11.5 %● |

1: Zone Files, Toplists, Blacklists (partially with NDA); 2: Excluding DNS names already included in Domain Lists; 3: Traceroute and ipmap data

★Amazon, ●Host Europe, ■Cloudflare, ▲Linode, ✣DTAG, ★ProXad, ●Hetzner, ■Comcast, ▲Swisscom, ✣Google, ★Antel, ●Versatel, ■BIHNET



(a) Cumulative runup of IPv6 addresses.

(b) AS distribution for hitlist sources.

(c) Hitlist addresses mapped to BGP prefixes.

Figure 1: Hitlist sources runup, AS distribution as CDF, and zesplot.

the Bitnodes API [57] that provides current peers of the Bitcoin network. Although this is the smallest source, contributing 27 k unique IPv6 addresses, we still find it valuable as it also adds client addresses.

**RIPE Atlas:** We extract all IPv6 addresses found in traceroutes of the RIPE Atlas project, as well as all IPv6 addresses from RIPE's ipmap project [43], which add another 0.2 M addresses. These are highly disjunct from previous sources, likely due to their nature as routers.

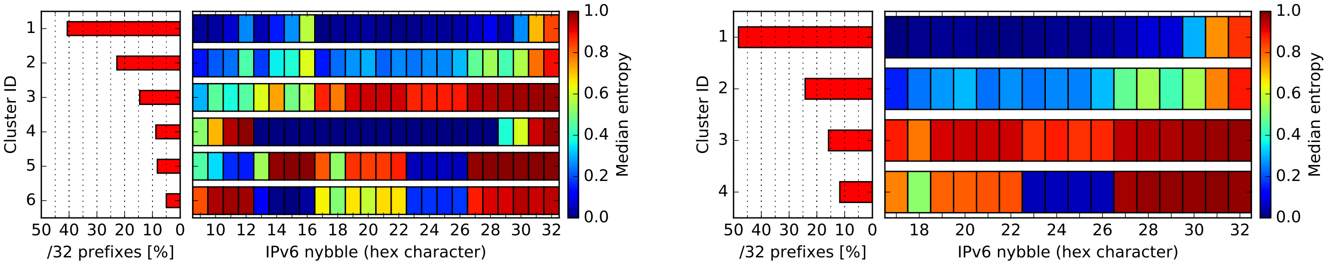
**Scamper:** Finally, we run traceroute measurements using scamper [32] on all scanning targets, and extract router IP addresses learned from these measurements. This source shows a very strong growth characteristic, with a total of 25.9M unique IP addresses. As we find this peculiar, we conduct a closer investigation, which reveals that 90.7% of those IP addresses are SLAAC addresses, *i.e.*, marked by `ff:fe`. The vendor codes in MAC addresses gained from those routers indicate that they are almost exclusively home routers: 47.9% ZTE and 47.7% AVM (Fritzbox), followed by 1.2% Huawei with a long tail of 240 other vendors. This shows that our source includes mainly home routers and CPE equipment, which may be undesirable to scan.

We accumulate all sources, *i.e.*, IP addresses will stay forever in our scanning list once added. We may revisit this decision in the future, and remove IP addresses after a certain window of unresponsiveness.

**Input Distribution:** When evaluating the AS distribution for each source in Figure 1b, we see stark differences, e.g., for domainlists and CT only a handful of ASes make up a large fraction of addresses, compared to the more balanced RIPE Atlas source. In addition, we analyze the distribution of hitlist addresses to BGP prefixes in Figure 1c. Although we achieve good coverage of BGP prefixes, we find that some prefixes contain unusually large numbers of addresses, which we investigate in Section 5.

**Comparison with DNSDB:** Comparing our hitlist with all AAAA records stored in DNSDB [21], we cover 12.9% IPv6 addresses, 69.4% ASes, and 48.7% BGP prefixes stored in DNSDB. The majority of "missing" addresses belong to large CDN operators, which is probably due to DNSDB collecting passive DNS data globally vs. active probing from a few locations. Moreover, the vast majority of addresses in our hitlist is not in DNSDB, especially for infrastructure of ISP operators. We find our hitlist and DNSDB to be complementary. We do not add DNSDB to our sources as it is not publicly available.

**Going Beyond:** Besides the aforementioned daily scanned sources, we also conduct in-depth case studies on three disjunct input sources: newly learned IPv6 addresses using 6Gen and Entropy/IP in Section 7, rDNS-walked IPv6 addresses in Section 8, and crowdsourced client IPv6 addresses in Section 9.



(a) Fingerprints of full addresses, $\mathbf{F}^9_{32}$

(b) Fingerprints of IIDs, $\mathbf{F}^{17}_{32}$

Figure 2: /32 prefixes clustered using entropy fingerprints.

# 4. ENTROPY CLUSTERING

We introduce a method for clustering IPv6 networks by similar entropy in their addresses and evaluate it on the IPv6 hitlist in the next subsection.

Let $S$ be a set of IPv6 addresses obtained for a particular network, e.g., a /32 prefix, a BGP prefix, or an AS. The set may be a random sample, but with at least 100 addresses. We introduce the following formal notation:

$$S = \{\ \mathbf{A}_1, \cdots, \mathbf{A}_i, \cdots, \mathbf{A}_n\ \},\ n \geq 100 \tag{1}$$

$$\mathbf{A}_i = (\ x^i_1, \cdots, x^i_j, \cdots, x^i_{32}\ ), \tag{2}$$

$$x^i_j \in \Omega = \{‘0‘, \cdots, ‘f‘\}, \tag{3}$$

where $\mathbf{A}_i$ is an address in that network: a sequence of 32 nybbles, *i.e.*, hex characters. Let $X_j$ be a discrete random variable on $\Omega$ representing nybble $j$ across $S$, and have an empirical probability mass function $\hat{P}(X_j)$. Next, we use the technique introduced in [24] to compute the normalized entropy of the nybbles, which we call an entropy fingerprint $\mathbf{F}^a_b$:

$$\mathbf{F}^a_b = (\ H(X_a), \cdots, H(X_j), \cdots, H(X_b)\ ) \tag{4}$$

$$H(X_j) = \frac{1}{4} \cdot - \sum_{\omega \in \Omega} \hat{P}(X_j = \omega) \cdot log\hat{P}(X_j = \omega). \tag{5}$$

where $a$ and $b$ are the first and the last considered nybble, respectively. If $H(X_j) = 0$, then nybble $j$ is constant; if $H(X_j) = 1$, then all its values are equally probable.

We repeat the above for each network and run the k-means algorithm on obtained dataset to find clusters of networks with similar fingerprints. We use the elbow method to find the number of clusters, $k$, plotting the sum of squared errors (SSE) for $k = \{1, \cdots, 20\}$:

$$SSE(k) = \sum_{c \in C_k} \sum_{\mathbf{E} \in c} d^2(\mathbf{E}, \bar{\mathbf{c}}) \tag{6}$$

where $C_k$ is the set of clusters obtained for given $k$, $\bar{\mathbf{c}}$ is the cluster mean, and $d^2$ is the squared Euclidean distance. We select the $k$ for which we see an "elbow" in the plot, *i.e.*, the point where increasing $k$ does not yield a relatively large reduction in $SSE$.

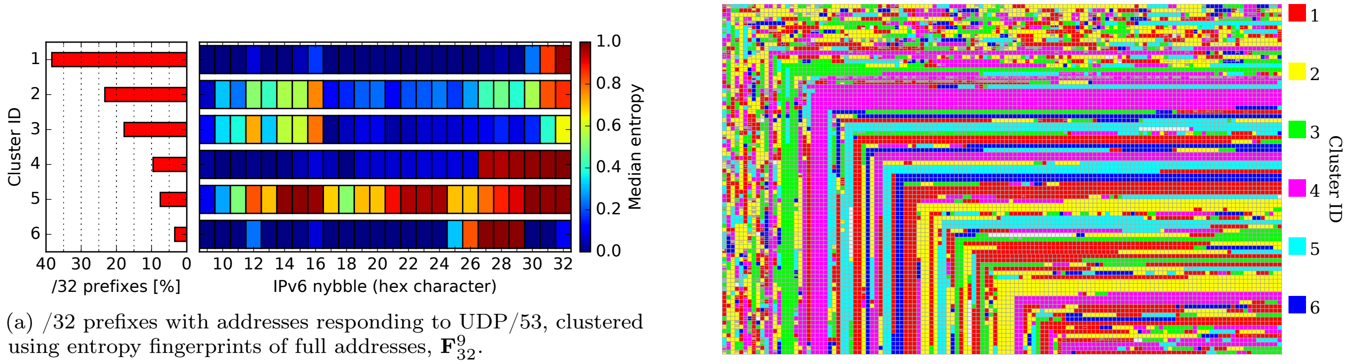
Finally, we summarize each cluster with its median entropy on each nybble and with its relative popularity.

## 4.1 Results

We present the results of applying entropy clustering on /32 prefixes in our hitlist in Figure 2.

Most notably, our method identifies just 6 clusters of full address fingerprints, as shown in Figure 2a. The most popular cluster has entropy close to 0 for all nybbles except for a few nybbles at the end of the network and interface identifiers. It is likely an effect of a common practice to treat both halves of IPv6 addresses as counters for networks and hosts. The second most popular pattern is similar, but making use of more nybbles and introducing more structure. In cluster 3, we see prefixes with pseudo-random interface identifiers (IIDs), as majority nybbles have high entropy, which makes predicting addresses in such networks impossible. Finally, we see MAC-based IIDs in clusters 5 and 6, where nybbles 23-26 are likely just `ff:fe`. In Figure 2b, we focus on IIDs only, *i.e.*, we limit fingerprints to nybbles 17-32, and find just 4 clusters. Again, majority of evaluated networks use IIDs as counters, as visible in clusters 1 and 2. This is expected and common, e.g., for pools of servers, but shows potential for probabilistic scanning. We see the impact of SLAAC addressing in clusters 3 and 4: pseudo-random and MAC-based, respectively.

Finally, we indicate the connection of entropy clustering to other methods and results presented later in this paper. In Figure 3a, we apply clustering to understand IPv6 scanning results detailed in Section 6. The figure shows clusters with addresses that respond to a UDP/53 (DNS) scan. The most popular cluster 1 has almost zero entropy on all but the last two nybbles, which makes probabilistic scanning for IPv6 DNS servers easy, which we demonstrate in Section 7. In Figure 3b, we show how clusters are distributed to BGP prefixes. In this plot, all plotted prefixes are still ordered by their prefix length and the AS announcing them, but the size of the box is static instead of based on the prefix length. This results in all rectangles equal in size, allowing for easier pattern spotting. With the ordering going from top-left to bottom-right, shorter (larger) prefixes in the top-left corner, we find that within the larger prefixes the mix of clusters is more heterogeneous than in the longer



(a) /32 prefixes with addresses responding to UDP/53, clustered using entropy fingerprints of full addresses, $\mathbf{F}^{9}_{32}$.

(b) BGP prefixes colored using their $\mathbf{F}^{9}_{32}$ clusters.

Figure 3: Entropy clustering for DNS responsive hosts and cluster distribution in BGP prefixes.

(smaller) prefixes. Going towards the longer prefixes in the bottom-right corner, more consistency is observed. Most of the large, equally colored chunks are equally sized prefixes in a single AS. This hints at operators using the same addressing scheme, or deploying equivalent equipment or setups, in all of their prefixes.

In summary, we believe the presented technique simplifies Internet studies. Instead of treating IPv6 as an opaque and immense addressing space, we can visualize actual addressing patterns and their popularity in one picture. Our results have implications for scanning the IPv6 Internet. On one hand, the clusters in Figure 2 suggest people in general use IPv6 addresses in a limited number of ways, and that many IPv6 address nybbles are easy to predict. On the other hand, cluster *popularities* represent our hitlist—and to some extent, IPv6 hitlists in general—rather than the Internet.

# 5. ALIASED PREFIX DETECTION

Detection of aliased network prefixes, *i.e.*, prefixes under which each possible IP address replies to queries, is a significant problem for IPv6 measurements. These areas can easily contribute vast numbers of addresses that map to the same server, e.g., through the `IP_FREEBIND` option in Linux. This feature is already in use by CDNs [33], and was identified as a challenge in previous works [22, 37]. Aliased prefixes can artificially inflate the number of IP addresses within a hitlist (e.g., enumerating a /96 prefix can add $2^{32}$ addresses), and introduce significant bias into any studies using these hitlists. Hence, we want to populate our hitlist only with valuable addresses, *i.e.*, belonging to different hosts and having balanced prefix and AS distributions. This requires reliable detection and removal of aliased prefixes, for which we introduce a rigorous method in the following.

Similar to [22, 37], our method roots in the concept that a randomly selected IP address in the vast IPv6 space is unlikely to respond. Hence, when probing randomly selected addresses, a prefix can be classified as aliased after a certain number of replies was received. Murdock *et al.* [37] send probes to three random addresses in each /96 prefix. Upon receipt of one reply, the prefix is determined as aliased. We improve the efficiency and effectiveness of that approach in several ways.

For alias detection to work at our scale, it needs to fulfill two criteria: (1) detection must be low-effort with a small number of packets required per network, (2) detection must function for end hosts, not routers, which excludes many alias detection techniques.

Table 3: IPv6 fan-out for multi-level aliased prefix detection example. We generate one pseudo-random address in `2001:0db8:407:8000::/64` for each of the 16 subprefixes, *i.e.*, `2001:0db8:407:8000:[0-f]/68`.

| 2001:0db8:0407:8000::/64 |
|---|
| 2001:0db8:0407:8000:0151:2900:77e9:03a8 |
| 2001:0db8:0407:8000:181c:4fcb:8ca8:7c64 |
| 2001:0db8:0407:8000:23d1:5e8e:3453:8268 |
| ⋮ |
| 2001:0db8:0407:8000:f693:2443:915e:1d2e |

## 5.1 Multi-Level Aliased Prefix Detection

For our daily scans, we perform multi-level aliased prefix detection (APD), *i.e.*, detection at different prefix lengths. This is in contrast to previous works that use static prefix lengths, e.g.,/96.

To determine whether a prefix is aliased, we send 16 packets to pseudo-random addresses within the prefix, using TCP/80 and ICMPv6. For each packet we enforce traversal of a subprefix with a different nybble. For example, to check if `2001:db8:407:8000::/64` is aliased, we generate 1 pseudo-random address for each of 4-bit subprefixes, `2001:db8:407:8000:[0-f]::/68`. See Table 3 for a visual explanation. Using this technique we ensure to (1) distribute probes evenly over more specific subprefixes and (2) target pseudo-random IP addresses, which are unlikely to respond. For each of the probed prefixes, we count the number of addresses that respond. If we obtain responses from all 16 probed addresses, we label the prefix as aliased.

We run the aliased prefix detection on both announced IPv6 prefixes in BGP and our hitlist. The former source allows us to understand the aliased prefix phenomenon on a global scale, even for prefixes where we do not have any targets. The latter source allows us to inspect our target prefixes more in-depth.

For BGP-based probing, we use each prefix as announced, without enumerating additional prefixes. For our hitlist, we map the contained addresses to all prefixes from 64 to 124, in 4-bit steps. As APD requires 32 probes to be sent per address, we only scan prefixes with more than 100 targets. We exempt /64 prefixes from this limitation to allow full analysis of all known /64 prefixes. We use 47.4 M probes to guarantee complete coverage of all /64 prefixes and 49.2 M probes in total.

As we perform target-based APD at several prefix lengths, the following four cases may occur:

1. Both more and less specific are aliased
2. Both more and less specific are non-aliased
3. More specific aliased, less specific non-aliased
4. More specific non-aliased, less specific aliased

The first two cases depict the "regular" aliased and non-aliased behaviors, respectively. The third case is more interesting as we observe divergent results based on the prefix length that we query. One example could be a /96 which is determined as being non-aliased, with only 9 out of 16 /100 subprefixes determined as aliased. This case underlines the need for our fan-out pseudo-random aliased prefix detection. Using purely random addresses, all 16 could by chance fall into the 9 aliased subprefixes, which would then lead to incorrectly labeling the complete /96 prefix as aliased. The fourth case is an anomaly, since an aliased prefix should not have more specific non-aliased subprefixes. One reason for this anomaly is packet loss for the subprefix probes, incorrectly labeling them as non-aliased.

We analyze how common the fourth case is in our results and investigate the reasons. On May 4, 2018, we detect only eight such cases at three prefix lengths: /80, /116, and /120, which we analyze in the following.

The /80 prefix shows 3 to 5 out of the 16 possible responses over time. The branches of responding probes differ between days, with no discernible pattern. We suspect this prefix is behind a SYN proxy, which is activated only after a certain threshold of connection attempts is reached. Once active, the SYN proxy responds to every incoming TCP SYN, no matter the destination.

The /116 prefix consistently shows 15 out of 16 probes being answered on consecutive days, even though a less specific prefix was classified as aliased. Moreover, the 15 probes answer with the same TCP options on consecutive days. The non-responding probe is always on the `0x0` branch, so we believe the subprefix is handled differently and not by an aliased system. In fact, comparing the paths of the different branches reveals that the `0x0` branch is answered by an address in a different prefix. The DNS reverse pointer of this address hints at a peering router at DE-CIX in Frankfurt, Germany. This /116 anomaly underlines the importance of the multi-level aliased prefix detection, since there are in fact small non-aliased subprefixes within aliased less specific prefixes.

The case of six neighboring /120 prefixes manifests less consistent than the previously described phenomenon. The branches that lack responses change from day to day, as well as from prefix to prefix. Subsequent manual measurements show that previously unresponsive branches become responsive. The root cause is most likely ICMP rate limiting, which explains the seemingly random responding branches. We try to counter packet and ICMP-rate limiting loss as explained in Section 5.2.

After the APD probing, we perform longest-prefix matching to determine whether a specific IPv6 address falls into an aliased prefix or not. This ensures we use the result of the most closely covering prefix for each IPv6 address, and creates an accurate filter for aliased prefixes. If a target IP address falls into an aliased prefix, we remove it from this day's ZMapv6 and scamper scans.

## 5.2 Loss Resilience

Packet loss might cause a false negative, *i.e.*, an aliased prefix being incorrectly labeled as non-aliased. To increase resilience against packet loss, we apply (1) cross-protocol response merging and (2) multi-day sliding window.

As we are probing all 16 target addresses on ICMPv6 and TCP/80, IP addresses may respond inconsistently. Our technique hinges on the fact that unlikely IP addresses respond at all, so we treat an address as responsive even if it replies only to one of the two probes. While this stabilizes our results immensely, we still see networks with high loss, which would fail automatic detection, but still could be confirmed manually as aliased.

To further tackle these, we introduce a sliding window over several past days, and require each IP address to have responded to any protocol in the past days. As prefixes may change their nature, we do this step very carefully, and aim for a very short sliding window to react to such changes as quickly as possible.

To find an optimum, we compare the number of days in the sliding window to the number of prefixes that are unstable, *i.e.*, change the nature of stable and unstable over several days. We show the data in Table 4, which confirms that with a sliding window of just 3 days, we can reduce the number of unstable prefixes by almost 80%, while only adding a small delay for prefixes that change their nature. With the finally selected sliding

Table 4: Impact of sliding window on the number of unstable prefixes.

| Sliding Window: | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Unstable Prefixes | 65 | 26 | 22 | 14 | 14 | 13 |

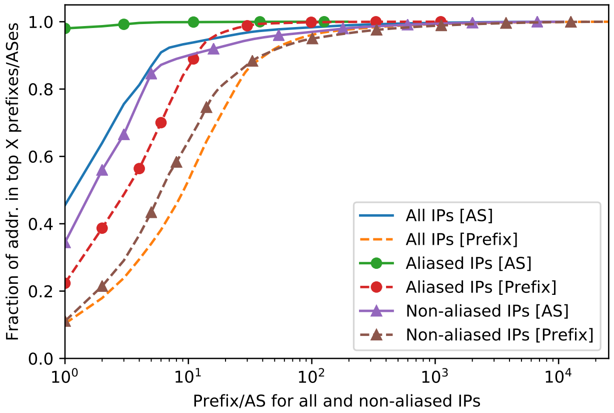


Figure 4: Prefix and AS distribution for aliased, non-aliased, and all hitlist addresses.

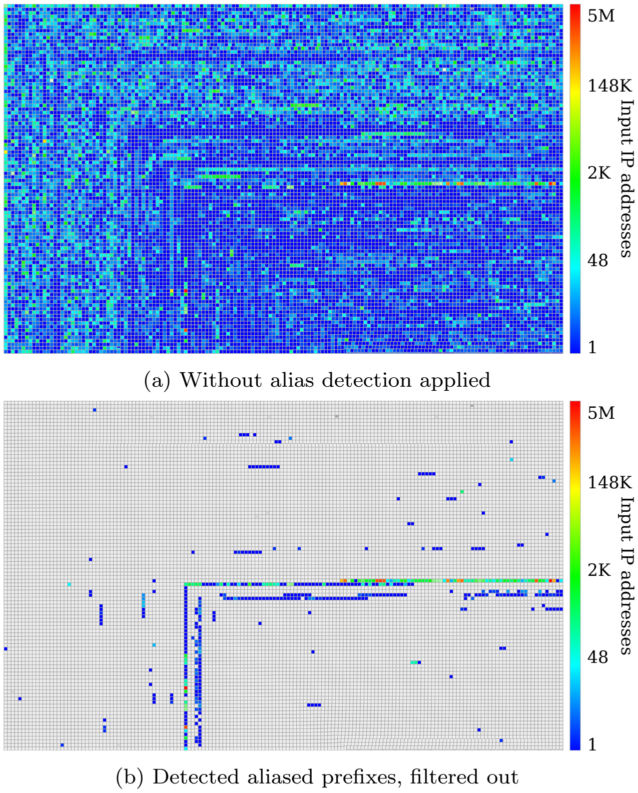
window of 3 days, only 14 of the 909 aliased prefixes as of May 1, 2018 show an unstable nature.

## 5.3 Impact of De-Aliasing

We apply the aliased prefix detection (APD) filtering daily and analyze the impact on our hitlist for May 11, 2018. Before filtering, there are a total of 55.1 M IPv6 addresses on our hitlist. After identifying aliased prefixes, 29.4 M targets (53.4 %) remain. With the unfiltered hitlist we cover 10,866 ASes and 25,465 announced prefixes. Removing aliased prefixes reduces our AS coverage by only 13 ASes, and lowers prefix coverage by 3.2 % to 24,648 prefixes.

We show the AS and prefix distribution for aliased, non-aliased, and all IPv6 addresses in Figure 4. By comparing AS distributions we find aliased prefixes as heavily centered around a single AS (Amazon). In consequence, the AS distribution for non-aliased prefixes is more flat than the entire population. The picture changes for prefix distributions: targets in non-aliased prefixes are now slightly more top-heavy compared to the general population. One of the reasons is that the vast majority of aliased IP addresses are within 189 /48 prefixes announced by Amazon, which are the shortest detected aliased prefixes. This results in shifting down the prefix distribution of the general population.

To further visualize the effect of filtering aliased prefixes, we compare Figure 5a with Figure 5b. In these figures, equally sized boxes are plotted for every prefix, ordered by prefix length and ASN. We find that aliasing barely occurs in the shortest prefixes, plotted in the top left corner of the plot. Towards the lower right end—*i.e.*, the longer prefixes—we find some groups of aliased /32 prefixes, but most eye-catching are the groups forming the big hook. These are all /48 prefixes, with the majority belonging to Amazon (the 'outer' hook) and Incapsula (the 'inner' hook). We also see how filtering these prefixes affects the number of probes we send: the brightly colored Amazon aliased prefixes comprise a large share of the input set.



(a) Without alias detection applied

(b) Detected aliased prefixes, filtered out

Figure 5: Responses on ICMP Echo requests

## 5.4 Fingerprinting Aliased Prefixes

The reason of not scanning aliased prefixes is that each contained IP address is assumed to belong to the same host, and consequently to display the same properties. We investigate this assumption by deploying fingerprinting techniques in our probes. Employing a ZMap module that supports the TCP Options header [45], we send a set of commonly supported fingerprinting options (`MSS-SACK-TS-WS`), using MSS and WS set to 1 to trigger differing replies [48].

Fingerprinting is a rather fuzzy technique with many challenges, which is why we carefully evaluate results and consider them indicative rather than conclusive. We consider this as a case study for validating our APD technique, and do not feed the results back to scanning.

Below, we investigate replies from 20,692 /64 prefixes classified as aliased, for which all of our 16 APD probes to TCP/80 succeeded on May 11th, 2018. We first analyze TTL values of response packets. Previous work found that TTL values cannot be expected to be constant per prefix or even IP address [8,47]. TTL inconsistencies can stem from routing changes, TTL-manipulating middleboxes, or other on-path effects. We quickly re-appraise this sentiment with our data, and can confirm that 5970 of our 20,692 prefixes offer inconsistent TTL values. We hence ignore the raw TTL metric and instead consider iTTL, which is the likely initial TTL value chosen by a host.

**iTTL:** Rounding the TTL value up to the next power of 2 results in the iTTL value [8,30,36]. Using iTTL, we find only 6 prefixes with inconsistent behavior, all caused by 22 IP addresses responding with differing iTTL values to our 2 consecutive probes. These addresses belong to only 2 /48 prefixes, announced by 2 unknown ASes. As the iTTL can be only one of four values (32, 64, 128, or 256), we use differing iTTL as a negative indicator for APD: many different iTTL values suggest a non-aliased prefix. With only 0.03 % of inconsistent prefixes, we remain confident in our APD filtering.

**Optionstext:** We next evaluate a metric used by [10, 48] that translates TCP options into a string, which preserves order and padding bytes, but not the values of options. For example, the string `MSS-SACK-TS-N-WS` would represent a packet that set the Maximum Segment Size, Selective ACK, Timestamps, a padding byte, and Window Scale options. Although we found 99.5% of responsive hosts to choose that option set, we identify 104 prefixes that send a differing set of TCP options, an unlikely behavior for an aliased machine.

**WScale and WSize:** The next metric is TCP window size and TCP window scale option. These options are also not necessarily expected to stay constant, as changes in host state can lead to advertising varying window sizes. However, a consistent window across all IP addresses in a prefix again raises our confidence in our technique. We find 1068 prefixes with inconsistent TCP window sizes, and 105 prefixes with inconsistent window scale options: a sizable number, but amounting to only ≈ 5% of our aliased prefixes.

**MSS:** Like iTTL, the TCP Maximum Segment Size is used as a negative indicator. Inconsistent MSS values are determined for 1030 prefixes. This behavior is, as for previous metrics, typically caused by individual IP addresses sending differing MSS sizes.

**Timestamps:** Finally, we evaluate TCP Timestamps, as suggested by [10,48]. Although TCP Timestamps offer highly discriminative features, they cannot reliably identify a prefix as non-aliased with only 2 probes per host, due to the variety of possible behaviors, e.g., randomized start values. They can, however, strengthen our confidence that an aliased prefix is behaving consistently. We hence run the following checks: (1) whether all hosts send the same (or missing) timestamps, (2) whether timestamps are monotonic for the whole prefix, and (3) whether receive timestamp and remote TCP timestamp have a regression coefficient $0.8 < R^2$. This tests for a global linear counter, strongly hinting the queried IP addresses belong to the same machine [10, 48]. If any of these three tests succeeds, we consider a prefix to offer consistent behavior, which is a strong indicator for aliasing.

Table 5: Fingerprinting 20.7 k aliased prefixes: inconsistent prefixes per test, in total, and total consistent.

| Test | Incs. | Σ Incs.. | Σ Cons. |
|---|---|---|---|
| iTTL | 6 | 6 | 20,686 |
| Optionstext | 104 | 110 | 20,581 |
| WScale | 105 | 215 | 19,515 |
| MSS | 1030 | 1175 | 19,513 |
| WSize | 1068 | 1186 | 19,506 |
| Timestamps | n/a[1] | n/a[1] | 13,202 |

[1] A failed timestamping test does not indicate an inconsistent, but an indecisive prefix

Table 6: Validation: aliased prefixes are considerably less inconsistent and pass many more consistency checks, including timestamping. Missing prefixes were indecisive.

| Scan Type | Incons. | Cons. |
|---|---|---|
| Non-Aliased Prefixes | 50.4% | 23.8% |
| Aliased Prefixes | 5.1% | 63.8% |

We find 13,202 of the 20,692 prefixes to exhibit a consistent behavior. Given that all Linux machines since kernel 4.10 would fail our tests as they randomize initial timestamps per `<SRC-IP, DST-IP>` tuple [48], we consider this a quite high indicator that our APD probing does indeed find aliased IP addresses. Note that due to many valid scenarios a failed timestamping test does not make a prefix inconsistent, but is simply indecisive as to whether a prefix may or may not be aliased.

Table 5 shows statistics for all performed consistency tests. Excluding TCP Timestamps, all tests combined find only 1186 inconsistent prefixes. Interestingly, >90% of those are caused by hosts showing surprisingly inconsistent behavior to our 2 probe packets. For example, we found 22 hosts responding with distinctively different iTTL values (64 vs. 255) in direct order. We also see hosts responding with different sets of TCP options or option values. Many of these are hosted at CDNs, and might be TCP-level proxies to other services, which could explain different fingerprints over time.

To verify our methodology, we run the same tests on *non*-aliased prefixes in our daily scans. For direct comparison, we only choose 2940 /64 prefixes with ≥16 responding IP addresses. As shown in Table 6, we find 1481 (50.4 %) to fail at least one of our tests, a considerably higher share than the 5.1 % among aliased prefixes. Additionally, we find only 699 prefixes (23 %) to pass our high-confidence test of similar timestamping behavior, compared to 63.8 % in aliased prefixes. This verification step shows that (1) our tests are discriminatory and (2) aliased prefixes offer far less diverse configurations, with many more cases believed to be the same machine.

## 6. ADDRESS PROBING

We generate IPv6 targets and probe these targets' responsiveness each day. First, we collect addresses from our hitlist sources. Second, we preprocess, merge, and shuffle these addresses in order to prepare them as input for scanning. Third, we perform aliased prefix detection to eliminate targets in aliased prefixes. Fourth, we traceroute all known addresses using scamper [32] to learn additional router addresses. Fifth, we use ZMapv6 [53] to conduct responsiveness measurements on all targets. We send probes on ICMP, TCP/80, TCP/443, UDP/53, and UDP/443 probes to cover the most common services [26]. We repeat this process each day to allow for longitudinal responsiveness analysis.

### 6.1 Responsive Addresses

We first evaluate responsive addresses based on their corresponding BGP prefix.

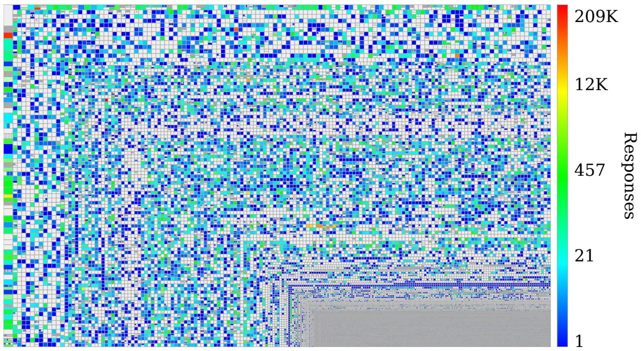


Figure 6: All announced prefixes, colored based on the number of responses to ICMP Echo requests on May 11 2018.

Overall, our hitlist contains 1.9 M responsive IPv6 addresses, spread over 21,647 BGP prefixes covering 9968 different ASNs.

Figure 6 shows non-aliased ICMP responsive addresses per BGP prefix. We see that most prefixes are covered with dozens to hundreds of responsive targets, whereas a few prefixes contribute 12 k or more responsive addresses. The plot is, in terms of colors, strikingly similar to the input set visualized in Figure 1c (note however that the range of the scale in the response plot is smaller). This tells us that for most of the prefixes in our input set, we indeed see responses, and only few return no responses at all. A possible explanation for prefixes with a sizable number of addresses in the input set ending up blank in the response plot, is dropping of ICMP echo requests at a border router.

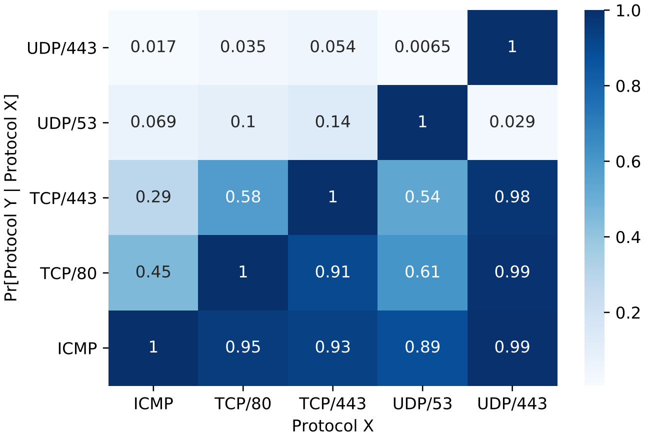


Figure 7: Conditional probability of responsivencess between services.

### 6.2 Cross-protocol Responsiveness

We analyze the cross-protocol responsiveness of our probes, to understand what kind of IPv6 hosts are responding to our probes. In Figure 7 we show the conditional probability of responsiveness between protocols, *i.e.*, if protocol X is responding, how likely is it that protocol Y will respond. We compare our findings for IPv6 to Bano *et al.* [9] who performed a similar analysis for IPv4.

We find that if an IPv6 address responds to one of the probes there is at least a 89 % chance of the same IP address also responding to ICMPv6. The ICMP correlation in IPv6 is higher compared to IPv4, where we see values as low as 73 %. Since ICMPv6 is an integral part of IPv6 it should not be simply blocked in firewalls [16], which makes it more likely that hosts are responding to ICMPv6 compared to its IPv4 counterpart.

Additionally, we see correlations between UDP/443, TCP/443, and TCP/80. More specifically, if an address is responsive to QUIC (UDP/443), it has a likelihood of 98 % to be also providing HTTPS and HTTP services. HTTPS servers are 91 % likely to provide an HTTP service as well, e.g., to offer a forwarding service to the secure version of a web page. Note that the reverse correlation (HTTP → HTTPS → QUIC) is far less pronounced. Compared to the HTTPS → HTTP correlation of 91 % in IPv6, we see only a 72 % correlation in IPv4 [9].

Analyzing DNS (UDP/53) correlation shows mostly similar results in IPv6 as in IPv4. One exception is the lower correlation probability to HTTPS (54 %) in IPv6 compared to the quite high one in IPv4 (78 %).

### 6.3 Longitudinal Responsiveness

To analyze address responsiveness over time, we probe an address continuously even if it disappears from our hitlist's input sources. We evaluate longitudinal respon-

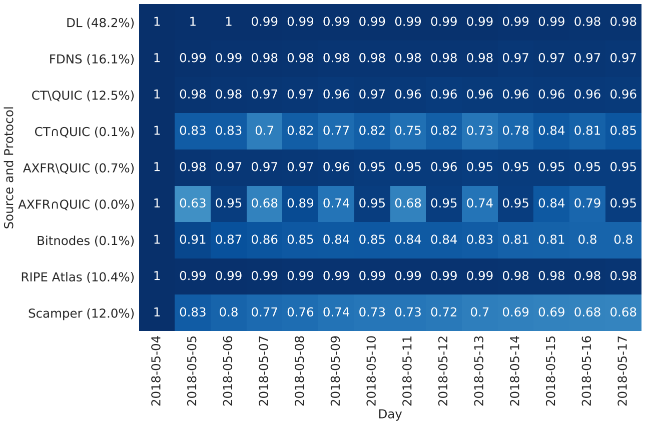


Figure 8: Responsiveness over time, split up by hitlist source and for special cases also probed protocol.

siveness over two weeks as depicted in Figure 8. As a baseline for each source we take all responsive addresses on the first day.

We find that IPv6 addresses from domain lists (DL), FDNS, and RIPE Atlas answer quite consistently over the 14 day period, with all three source losing only a few percentages of addresses. CT and AXFR sources overall reach similarly stable response rate; their QUIC response rates, however, fluctuate more heavily and are therefore depicted separately. Sources which include clients or CPE devices such as Bitnodes and Scamper lose 20 % and 32 % of the responding hosts, respectively.

# 7. LEARNING NEW ADDRESSES

In addition to acquiring IPv6 addresses through domain names and other sources, we can also detect addressing schemes and leverage those patterns to learn previously unknown addresses.

## 7.1 Methodology

To generate previously unknown addresses we feed our hitlist into a re-implementation of Entropy/IP [24], and a pre-release version of 6Gen [37]. For this work, we improve the address generator of Entropy/IP so that it walks the Bayesian network model exhaustively instead of randomly. We believe the improved generator lets us focus on more probable IPv6 addresses, under a constrained scanning budget.

First, we use all addresses in non-aliased prefixes to build a seed address list. Excluding aliased prefixes avoids generating addresses in prefixes where all addresses are responsive, and thus artificially distorting the response rate. Second, we split the seed address list based on ASes, as we assume similar addressing patterns within the same AS. We limit the eligible ASes to those with at least 100 IPv6 addresses to increase the probability of 6Gen and Entropy/IP identifying patterns. Third, we take a random sample of at most 100 k IPv6 addresses per AS to use as input for 6Gen and Entropy/IP. The capped random sample ensures that we provide a balanced input for each AS. Fourth, we run Entropy/IP and 6Gen with the capped random sample as input to generate 1 M addresses for each AS separately. Fifth, we again take a random sample of at most 100 k of all generated addresses per AS for 6Gen and Entropy/IP, respectively. The capped random sample ensures that ASes with more generated addresses are not overrepresented. Sixth, we perform active measurements to evaluate the value of the generated addresses.

## 7.2 Learned Addresses

Entropy/IP generates 118 M addresses. Of those, 116 M are routable and new addresses not yet in our hitlist. 6Gen produces slightly more addresses, 129 M, of which 124 M are new and routable. In total, we learn 239 M new unique addresses. Interestingly, there is very little overlap between 6Gen's and Entropy/IP's generated addresses: only 675 k addresses are produced by both tools, which equals to 0.2 % of all generated addresses.

## 7.3 Responsiveness of Learned Addresses

We probe the responsiveness of all 239 M learned addresses on ICMP, TCP/80, TCP/443, UDP/53, and UDP/443. 785 k IPv6 addresses respond to our probes, which corresponds to a response rate of 0.3 %. This low response rate underlines the challenges of finding new responsive addresses through learning-based approaches.

Comparing the responsiveness of addresses generated by 6Gen to Entropy/IP, we find that 6Gen is able to find almost twice as many responsive addresses: 489 k vs. 278 k. In addition, both tools found the same 17 k responsive addresses. The response rate of overlapping addresses generated by both 6Gen and Entropy/IP is therefore 2.5 %, which is an order of magnitude higher than the general learned address population's 0.3 %. This demonstrates that Entropy/IP and 6Gen find complementing sets of responsive IPv6 addresses, with a small overlap of targets that are more likely to respond. Thus, it is meaningful to run multiple address generation tools even on the same set of input addresses.

Table 7: Top 5 responsive protocol combinations for 6Gen and Entropy/IP.

| ICMP | TCP/80 | TCP/443 | UDP/53 | UDP/443 | 6Gen | Entropy/IP |
|---|---|---|---|---|---|---|
| ✓ | ✗ | ✗ | ✗ | ✗ | 66.8 % | 41.1 % |
| ✓ | ✓ | ✓ | ✗ | ✗ | 9.2 % | 12.3 % |
| ✗ | ✗ | ✗ | ✓ | ✗ | 7.3 % | 23.1 % |
| ✓ | ✓ | ✗ | ✗ | ✗ | 4.9 % | 3.4 % |
| ✓ | ✓ | ✓ | ✗ | ✓ | 3.2 % | 6.1 % |

When analyzing the top 5 protocols for responsive learned addresses in Table 7, we find particular differences between 6Gen and Entropy/IP. Two thirds of 6Gen responsive addresses answer to ICMP only, which is the

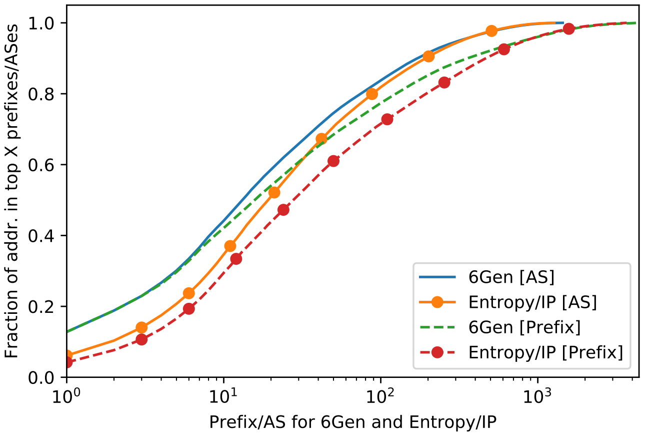


Figure 9: Prefix and AS distribution for responsive addresses generated with 6Gen and Entropy/IP.

case for only four out of ten Entropy/IP responsive addresses. On the other hand, Entropy/IP responsive hosts are three times more likely to be DNS servers (UDP/53). Moreover, 6Gen responsive hosts are half as likely to be QUIC-enabled web servers (ICMP, TCP/80, TCP/443, and UDP/443) compared to Entropy/IP. This shows that 6Gen and Entropy/IP not only discover mostly non-overlapping addresses, but also different types of *populations* of responsive hosts.

Finally, we compare ASes and prefixes of responsive addresses for both tools. 6Gen discovers responsive hosts in 1442 ASes, while Entropy/IP in 1275 ASes. Interestingly, responsive hosts in 384 ASes are found only by one of the tools, i.e., either 6Gen or Entropy/IP. In Figure 9, we show the prefix and AS distributions of responsive hosts. Entropy/IP's distribution is a bit less top-heavy compared to 6Gen's, where the top 2 responsive ASes make up almost 20 % of all addresses. Although there is some overlap in the top 5 ASes, 6Gen features more ISPs, like Sky Broadband, Google Fiber, and Xs4all Internet. In contrast, Entropy/IP's top ASes contain more CDNs and Internet services.

To summarize, 6Gen and Entropy/IP find few overlapping responsive addresses, but mostly in overlapping ASes. The services offered by these hosts differ considerably. Therefore, both tools have their advantages in finding specific addresses and populations. We suggest to run both tools to maximize the number of found responsive addresses. To optimize the response rate of generated addresses, one can probe only the addresses produced by both tools, as these are ten times more likely to respond.

## 8. RDNS AS A DATA SOURCE

In addition to the sources described in Section 3, we investigate the usefulness of IPv6 rDNS entries for active measurements. As shown by previous work, rDNS walking can be a source for IPv6 addresses [22,23]. While IPv6 rDNS addresses were used to, e.g., find mis-configured IPv6 networks [11], we are not aware of studies evaluating overall responsiveness. Since walking the rDNS tree to harvest IPv6 addresses is a large effort and puts strain on important Internet infrastructure, we classify this source as "semi-public", compared with sources such as the Alexa Top 1M list, which is available for download.

We use IPv6 rDNS data provided by Fiebig *et al.* [23] to perform active measurements and compare the results against other hitlist sources. Analyzing the overlap and structure of IPv6 addresses obtained from rDNS, we find a very small intersection with our hitlist. Of the 11.7 M addresses from rDNS, 11.1 M are new. The prefix distribution of rDNS and hitlist addresses are quite similar, as shown in Figure 10. The AS distribution is even more balanced for rDNS addresses compared to the hitlist. Therefore the addition of rDNS data to the hitlist input would not introduce prefix or AS bias.

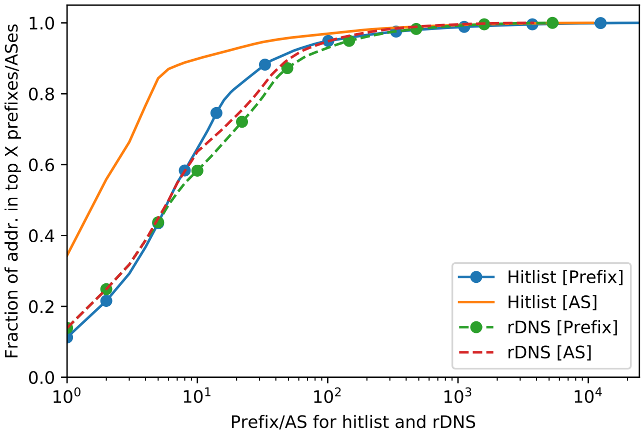


Figure 10: Prefix and AS distribution for hitlist and rDNS input data.

Next, we perform active measurements to compare the response rate of the rDNS population to the hitlist population. Before the active measurement, we filter 2.1 M unrouted addresses and 13.1 k addresses residing in aliased prefixes (see Section 5) from the rDNS addresses. The response rate for the hitlist with only non-aliased prefixes is generally similar to the response rate of rDNS. The rDNS ICMP response rate is higher with 10 % compared to the hitlist's 6 %. On the other hand, we receive slightly fewer HTTP(S) responses for rDNS with 2 % (1 %) against the hitlist's 3 % (2 %).

To ensure that responding rDNS addresses are not mostly client addresses, we first analyze the top ASes. As can be seen in Table 8, the top responsive ASes in the rDNS data are hosting and service providers, *i.e.*, mostly servers (especially in the TCP/80 measurement). Next, we look for IPv6 SLAAC's distinct `ff:fe` sequence and evaluate the hamming weight of IIDs for responsive

Table 8: Top 5 rDNS ASes in input and responsive via ICMP and TCP/80.

| # | Input | | ICMP | | TCP/80 | |
|---|---|---|---|---|---|---|
| 1 | Comcast | 12.5 % | Online S.A.S. | 19.6 % | Google | 12.8 % |
| 2 | AWeber | 10.2 % | Sunokman | 17.8 % | Hetzner | 10.1 % |
| 3 | Yandex | 9.8 % | Latnet Serviss | 8.7 % | Freebit | 6.8 % |
| 4 | Belpak | 6.2 % | Yandex | 7.9 % | Sakura | 6.5 % |
| 5 | Sunokman | 6.1 % | Salesforce | 5.3 % | TransIP | 5.0 % |

rDNS addresses, as an additional indicator for clients. We find between 6 % and 9 % SLAAC addresses with `ff:fe`. The IID hamming weight (*i.e.*, number of bits set to 1) can be used to infer the presence of clients with privacy extensions enabled [26]. The rDNS IID hamming weight does not suggest to contain a large client population, especially for TCP/80, where 60 % of addresses have a hamming weight of six or smaller.

To conclude, the responsive part of the rDNS data source does not add a client-heavy population and still keeps our focus on IPv6 enabled servers. We therefore suggest to add rDNS data as input to the IPv6 hitlist.

# 9. CLIENT IPV6 ADDRESSES

The majority of IPv6 addresses that we have collected so far belong to servers or routers rather than clients. In this section, we use crowdsourcing platforms to collect client IP addresses. Our aim is to start studying the following questions: Does crowdsourcing serve as an adequate source of residential IPv6 addresses, and might we be able to use addresses gathered through crowdsourcing in IPv6 hitlists? We refer the reader to Section 10.1 describing the ethical consideration of this study.

## 9.1 Experimental Setup

To investigate our approach of collecting IPv6 client addresses, we use two crowdsourcing platforms and leverage the *test-ipv6.com* [19] code to gather IPv6 addresses.

We set up a webserver and integrate it into the crowdsourcing environment to operate similarly to the study by Huz *et al.* [29]. We use Amazon Turk (Mturk) [5] and Prolific Academic (ProA) [41] to run our experiments and allocate a budget of 150 $ per platform. We add a limitation for only one submission per user per platform and select the minimum amount of money that can be set for each assignment: 0.01 $ for Mturk and 0.12 $ for ProA. We run the experiment between April 23 and May 23, 2018. During this time, 5781 users participate from Mturk, compared to 1186 from ProA.

## 9.2 Crowdsourcing Participants

Table 9 shows the distribution of measurements per platform, including AS [7] and country mappings [35]. We find about 31 % of Mturk and 20.6 % of ProA users with IPv6 enabled. One reason for Mturk's higher IPv6 ratio might be their customer base: Mturk is more popular in US and India [29, 31] both countries with considerable IPv6 adoption [28].

Moreover, 31.5 % of IPv6 ASes are overlapping between platforms, although we do not find any common addresses.

A large part of our IPv6 clients participate from a small number of ISPs. The top 3 ASes are the two US-based ISPs Comcast (31.1 %) and AT&T (13.2 %), and the Indian ISP Reliance (7.8 %). Comparing IPv6 clients with IPv4 clients we find the latter to be more diverse, where the top 5 providers constitute only 30 % of all IPv4 clients in our dataset.

Table 9: Client distribution in crowdsourcing study.

| # | IPv4 | IPv6 | $\text{ASN}_4$ | $\text{ASN}_6$ | $\#\text{cc}_4$ | $\#\text{cc}_6$ |
|---|---|---|---|---|---|---|
| Mturk | 5,707 | 1,787 | 842 | 73 | 93 | 22 |
| ProA | 1,176 | 245 | 272 | 48 | 33 | 21 |
| Unique | 6,862 | 2,032 | 983 | 92 | 98 | 29 |

## 9.3 Client Responsiveness

Once the user submits the results, we send an ICMP echo request and traceroute to each IPv6 address every 5 minutes.

We find that only 352 (17.3 %) of IPv6 addresses respond to at least one ICMP echo requests. The majority of IPv6 addresses we gathered from residential networks do not respond to ICMP echo requests.

To investigate whether the low response rate was an artifact of the devices (e.g., privacy extension address cycling, users disconnecting), or network policy, we locate the set of RIPE Atlas probes situated in the same ASes as our crowdsourced client addresses. RIPE Atlas probes will respond to echo requests they receive.

We select 1398 RIPE Atlas probes with IPv6 connectivity inside these ASNs and, having confirmed they were online, send traceroutes to those probes. As many as 641 (45.8 %) probes respond to our ping messages. Since RIPE Atlas probes will respond by design, these 45.8 % indicate an upper bound of possible crowdsourcing responses. It is likely that users' systems are running local firewalls which will reduce the response rate further. This is indicative only, and deserves further study.

We also find that for 20 % of clients, the last hop that responded is different from a destination AS. That further indicates inbound filtering by ISPs.

Client addresses that respond to our ICMP requests are likely to be less stable than servers' IP addresses. Only 7 IPv6 addresses from the total of 352 responsive addresses remain active for the entire period of the study. 19 % of IPv6 addresses are active for less than an hour, while 39.4 % of addresses are active for 8 hours or less. Moreover, addresses with dynamic behaviour *i.e.*, appearing and disappearing multiple times during the study had an average uptime of aproximately 8 hours and median uptime of 3 hours per day.

We conclude that crowdsourced addresses provide additional targets for IPv6 client studies. Only a fraction of them are, however, responsive to incoming probes. Finally, active measurements targeting crowdsourcing addresses need to be performed swiftly after address collection, as the responsive client population is quickly shrinking.

# 10. MEASUREMENT PRACTICES

In our study we follow best measurement practices by conducting scans in an ethical way and publishing data and code for reproducibility in research.

## 10.1 Ethical Considerations

Before conducting active measurements we follow an internal multi-party approval process, which incorporates proposals by Partridge and Allman [38] and Dittrich *et al.* [17]. We assess whether our measurements can induce harm on individuals in different stakeholder groups. As we limit our query rate and use conforming packets, it is unlikely for our measurements to cause problems on scanned systems. We follow best scanning practices [18] by maintaining a blacklist and using dedicated servers with informing rDNS names, web sites, and abuse contacts. During our six months of active scans, we received four emails asking for more information on the conducted measurements.

Prior to deploying the crowdsourcing study we discussed the subject with the university ethics committee. We agreed with the committee that collected IPv6 addresses will not be published. In addition, we informed participants that they could opt out from the active probing. The ethics committee concluded that our experiments are not human subjects research, as we have no reasonable way to map collected IPv6 addresses to individuals, and gave formal permission to conduct the study.

## 10.2 Reproducible Research

To encourage reproducibility in network measurement research [1, 50], we plan to make data, source code, and analysis tools publicly available. This includes tools such as zesplot and Entropy/IP, interactive zesplot graphs, and results from daily runs of the IPv6 hitlist, including the list of aliased prefixes. Some of the figures in this paper are clickable, offering additional insights, in-depth analyses, and interactive graphs. This data can serve as a valuable starting point for future IPv6 studies.

# 11. CONCLUSION

In this work we leveraged a multitude of source to create the largest IPv6 hitlist to date, containing more than 50 M addresses. We found that about half of these addresses reside in aliased prefixes and showed using clustering that there are only six prevalent IPv6 addressing schemes. Using longitudinal measurements we identified protocols and sources which are less stable over time. We used and extended state of the art tools to generate new addresses, finding that they provide complementary address sets. We will keep running daily IPv6 measurements to provide valuable hitlists to researchers.

**Acknowledgments:** We thank Austin Murdock for providing a pre-release version of 6Gen and Tobias Fiebig for the IPv6 rDNS data.

# 12. REFERENCES

[1] ACM. Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-badging.

[2] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman. Zippier ZMap: Internet-Wide Scanning at 10 Gbps. In *WOOT*, 2014.

[3] R. Almeida, O. Fonseca, E. Fazzion, D. Guedes, W. Meira, and Í. Cunha. A Characterization of Load Balancing on the IPv6 Internet. In *Passive and Active Measurement Conference*, 2017.

[4] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz. Mission Accomplished? HTTPS Security after DigiNotar. In *ACM Internet Measurement Conference*, 2017.

[5] Amazon. Mechanical Turk. https://www.mturk.com/.

[6] APWG. APWG: Cross-industry Global Group Supporting Tackling the Phishing Menace. http://antiphishing.org.

[7] H. Asghari. PyASN Python module. https://github.com/hadiasghari/pyasn.

[8] M. Backes, T. Holz, C. Rossow, T. Rytilahti, M. Simeonovski, and B. Stock. On the Feasibility of TTL-based Filtering for DRDoS Mitigation. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, 2016.

[9] S. Bano, P. Richter, M. Javed, S. Sundaresan, Z. Durumeric, S. J. Murdoch, R. Mortier, and V. Paxson. Scanning the Internet for Liveness. *SIGCOMM Computer Communication Review*, 48(2), May 2018.

[10] R. Beverly and A. Berger. Server Siblings: Identifying Shared IPv4/IPv6 Infrastructure via Active Fingerprinting. In *Passive and Active Measurement Conference*, 2015.

[11] K. Borgolte, S. Hao, T. Fiebig, and G. Vigna. Enumerating Active IPv6 Hosts for Large-scale Security Scans via DNSSEC-signed Reverse Zones. In *IEEE Symposium on Security and Privacy*, 2018.

[12] M. Bruls, K. Huizing, and J. J. Van Wijk. Squarified treemaps. In *Data visualization*. 2000.

[13] R. Bush, J. Hiebert, O. Maennel, M. Roughan, and S. Uhlig. Testing the reachability of (new) address space. In *SIGCOMM Workshop on Internet Network Management*, 2007.

[14] Censys. ICMP Echo Request Full IPv4 Scan Results. https://censys.io/data/0-icmp-echo_request-full_ipv4.

[15] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet mapping: from art to science. In *Conference For Homeland Security. Cybersecurity Applications & Technology*, 2009.

[16] E. Davies and J. Mohacsi. Recommendations for Filtering ICMPv6 Messages in Firewalls. RFC 4890 (Informational), May 2007.

[17] D. Dittrich et al. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. *US DHS*, 2012.

[18] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security*, 2013.

[19] Falling-Sky project. Test your IPv6. https://github.com/falling-sky/source/wiki.

[20] X. Fan and J. Heidemann. Selecting representative IP addresses for Internet topology studies. In *ACM Internet Measurement Conference*, 2010.

[21] Farsight Security. DNSDB. https://www.farsightsecurity.com/solutions/dnsdb/.

[22] T. Fiebig, K. Borgolte, S. Hao, C. Kruegel, and G. Vigna. Something from Nothing (There): Collecting Global IPv6 Datasets from DNS. In *Passive and Active Measurement Conference*, 2017.

[23] T. Fiebig, K. Borgolte, S. Hao, C. Kruegel, G. Vigna, and A. Feldmann. In rDNS We Trust: Revisiting a Common Data-Source's Reliability. In *Passive and Active Measurement Conference*, 2018.

[24] P. Foremski, D. Plonka, and A. Berger. Entropy/IP: Uncovering Structure in IPv6 Addresses. In *ACM Internet Measurement Conference*, 2016.

[25] O. Gasser, B. Hof, M. Helm, M. Korczynski, R. Holz, and G. Carle. In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements. In *Passive and Active Measurement Conference*, 2018.

[26] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle. Scanning the IPv6 Internet: Towards a Comprehensive Hitlist. In *Workshop on Traffic Monitoring and Analysis*, 2016.

[27] F. Gont and T. Chown. Network Reconnaissance in IPv6 Networks. RFC 7707 (Informational), Mar. 2016.

[28] Google. Per-Country IPv6 adoption. https://www.google.com/intl/en/ipv6/statistics.html.

[29] G. Huz, S. Bauer, R. Beverly, et al. Experience in using MTurk for Network Measurement. In *SIGCOMM C2B(I)D Workshop*, 2015.

[30] C. Jin, H. Wang, and K. G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic. In *ACM Computer and Communications Security Conference*, 2003.

[31] Q. Lone, M. Luckie, M. Korczynski, H. Asghari, M. Javed, and M. van Eeten. Using Crowdsourcing Marketplaces for Network Measurements: The Case of Spoofer. In *Network Traffic Measurement and Analysis Conference*, 2018.

[32] M. Luckie. Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet. In *ACM Internet Measurement Conference*, 2010.

[33] M. Majkowski. Abusing Linux's firewall: the hack that allowed us to build Spectrum. https://blog.cloudflare.com/how-we-built-spectrum/.

[34] Matthew Bryant. TLD AXFR transfers. https://github.com/mandatoryprogrammer/TLDR.

[35] Maxmind. GeoLite 2 database. https://www.maxmind.com/en/home.

[36] A. Mukaddam, I. Elhajj, A. Kayssi, and A. Chehab. IP Spoofing Detection Using Modified Hop Count. In *Advanced Information Networking and Applications Conference*, 2014.

[37] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson. Target Generation for Internet-wide IPv6 Scanning. In *ACM Internet Measurement Conference*, 2017.

[38] C. Partridge and M. Allman. Ethical Considerations in Network Measurement Papers. *Communications of the ACM*, 2016.

[39] PhishTank. A Nonprofit Anti-phishing Organization. http://www.phishtank.com.

[40] D. Plonka and A. W. Berger. kIP: a Measured Approach to IPv6 Address Anonymization. http://arxiv.org/abs/1707.03900, 2017.

[41] Prolific. Prolific Academic. https://www.prolific.ac/.

[42] Rapid7 Project Sonar. Forward DNS Data. https://opendata.rapid7.com/sonar.fdns_v2/.

[43] RIPE NCC. IPMap. https://ftp.ripe.net/ripe/ipmap/.

[44] Robert Graham. MASSCAN: Mass IP port scanner, 2013.

[45] Q. Scheitle. TCP Options module for ZMap. https://github.com/tumi8/zmap/blob/master/src/probe_modules/module_tcp_synopt.h.

[46] Q. Scheitle, T. Chung, J. Hiller, O. Gasser, J. Naab, R. van Rijswijk-Deij, O. Hohlfeld, R. Holz, D. Choffnes, A. Mislove, and G. Carle. A First Look at Certification Authority Authorization (CAA). *ACM SIGCOMM Computer Communication Review*, 2018.

[47] Q. Scheitle, O. Gasser, P. Emmerich, and G. Carle. Carrier-Grade Anomaly Detection Using Time-to-Live Header Information. http://arxiv.org/abs/1606.07613, 2016.

[48] Q. Scheitle, O. Gasser, M. Rouhi, and G. Carle. Large-scale Classification of IPv6-IPv4 Siblings with Variable Clock Skew. In *Network Traffic Measurement and Analysis Conference*, 2017.

[49] Q. Scheitle, O. Gasser, P. Sattler, and G. Carle. HLOC: Hints-Based Geolocation Leveraging Multiple Measurement Frameworks. In *Network Traffic Measurement and Analysis Conference*, 2017.

[50] Q. Scheitle, M. Wählisch, O. Gasser, T. C. Schmidt, and G. Carle. Towards an Ecosystem for Reproducible Research in Computer Networking. In *ACM SIGCOMM Reproducibility Workshop*, 2017.

[51] Spamhaus. The Spamhaus Project. www.spamhaus.org.

[52] S. D. Strowes. Bootstrapping Active IPv6 Measurement with IPv4 and Public DNS. http://arxiv.org/abs/1710.08536, 2017.

[53] TUM. ZMapv6 on GitHub. https://github.com/tumi8/zmap.

[54] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl. On reconnaissance with IPv6: a pattern-based scanning approach. In *Availability, Reliability and Security Conference*, 2015.

[55] P. van Dijk. Finding v6 hosts by efficiently mapping ip6.arpa. https://web.archive.org/web/20170603234058/http://7bits.nl/blog/posts/finding-v6-hosts-by-efficiently-mapping-ip6-arpa, Mar. 2012.

[56] M. Varvello, J. Blackburn, D. Naylor, and K. Papagiannaki. EYEORG: A Platform For Crowdsourcing Web Quality Of Experience Measurements. In *ACM Conference on Emerging Networking EXperiments and Technologies*, 2016.

[57] A. Yeow. Bitnodes API.
https://bitnodes.earn.com/.