Something From Nothing (There): Collecting Global IPv6 Datasets From DNS

Tobias Fiebig[1], Kevin Borgolte[2], Shuang Hao[2],
Christopher Kruegel[2], Giovanni Vigna[2]

[1]TU Berlin
[2]UC Santa Barbara

Abstract. Current large-scale IPv6 mostly use non-public datasets, as most public datasets are domain specific, e.g., traceroute-based datasets are biased toward network equipment. In this paper, we present a new methodology to collect IPv6 address datasets that does not require access to restricted network vantage points. We collect a new dataset spanning more than 5.8 million IPv6 addresses by exploiting DNS' denial of existence semantics (NXDOMAIN). This paper documents our efforts in a reproducible manner, so others can avoid the obstacles we encountered, and to obtain new datasets of allocated IPv6 addresses.

1 Introduction

The adoption of IPv6 has been steadily increasing in recent years [**?**]. Unsurprisingly, simultaneously, the research question of efficiently identifying allocated IPv6 addresses has received more and more attention from the scientific community. However, unfortunately for the common researcher these studies have—so far—been dominated by the analysis of large, restricted, and proprietary datasets. For instance, the well-known content delivery network (CDN) dataset used for most contemporary IPv6 analyses [14, 8], Internet exchange point (IXP) datasets which were used regularly by some other research groups [3, 9], or, slightly less restrictive, the Farsight DNS recursor dataset [20]. Although public datasets do exist, they are traceroute-based datasets from various sources, including the RIPE Atlas project [16], which are limited due to their nature: they are biased towards addresses of networking equipment, and, in turn, bear their own set of problems for meaningful analyses.

Correspondingly, in this paper, we aim to tackle the problem of obtaining a dataset of currently-allocated IPv6 addresses for the common researcher: We present a new methodology with which we were able to collect more than 5.8 million unique IPv6 addresses that can be employed by every researcher with network access. The underlying concept is the enumeration of IPv6 reverse zones (PTR) leveraging the semantics of DNS' denial of existence records (NXDOMAIN). Although the general concept has been discussed in RFC 7707 [10], we identified and overcame various challenges that prevent the use of this technique on a global scale. Therefore, we document how we can leverage the semantics of NXDOMAIN on a global scale to collect allocated IPv6 addresses for a new IPv6 dataset. Our detailed algorithmic documentation allows researchers everywhere to implement this technique, reproduce our results, and collect similar datasets for their own research.

In this paper, we make the following contributions:

- We present a novel methodology to enumerate allocated IPv6 addresses *without* requiring access to a specific vantage point, e.g., a CDN, IXP, or large transit provider.
- We focus on the reproducibility of our techniques and tools, to provide researchers with the opportunity to collect similar datasets for their own research.
- We report on a first set of global measurements using our technique, in which we gather a larger and more diverse dataset that provides new insights into IPv6 addressing.
- We present a case-study that demonstrates how our technique allows insights into operators' networks that could not be accomplished with previous techniques.

2 Previous Work

Active probing for network connected systems is probably one of the oldest techniques on the Internet. However, tools that can enumerate the full IPv4 space are relatively new. The first complete toolchain that allowed researchers to scan the whole IPv4 space was presented by Durumeric in 2013 [6] with ZMap. The problem of scanning the whole IPv4 address space is mostly considered solved since then. Especially the security scene heavily relies on these measures since then [18]. The search space for IPv6 is with 128bit, significantly larger than the 32bit of IPv4. Hence, a simple brute-force approach as presented for IPv4 is—so far—not feasible. Indeed, most current research efforts in the networking community are concerned with evaluating large datasets to provide descriptive information on utilized IPv6 addresses [10].

Plonka and Berger provide a first assessment of active IPv6 addresses in their 2015 study using a large CDN's access statistics as dataset [14]. Subsequently, in their 2016 work [8] Foremski et al. propose a technique to generate possibly utilized IPv6 addresses from initial seed datasets for later active probing. Gasser et al. attempt a similar endeavor, using among various other previously mentioned datasources a large Internet Exchange Point (IXP) as vantage point [9]. However, these works have the drawback that the used vantage points are not publicly available. Measurement-studies using public data sources have been recently published by Czyz et al. [**?**,5]. In their works they combine various public data sources like the Alexa Top 1,000,000 and the Farsight DNS recursor dataset [20]. In addition they resolve all IPv4 reverse pointers and then attempt to resolve the returned FQDNs for their IPv6 address.

3 DNS Enumeration Techniques

Complimentary to prior approaches, van Dijk proposed enumerating IPv6 reverse records by utilizing the specific semantics of denial of existence records (NXDOMAIN) [10, **?**]: When correctly implementing RFC1034 [**?**], as clarified in RFC8020 [**?**], the Name Error response code (NXDOMAIN in practice) has the semantic of *there is nothing* ***here*** *or anywhere* ***thereunder*** *in the name tree*. Making this notion explicit in RFC8020 [**?**] is a relatively recent development. Combined with the IPv6 PTR DNS tree, where each sub-zone has 16 (0-f, one for each IPv6 nibble) children up to a depth of 32 levels, provides the possibility to exploit standard-compliant nameservers to enumerate the zone.

Algorithm 1: Algorithm for iterating over ip6.arpa., based on RFC7707 [10].

```
// Base-Case: $max.ip6.arpa.len = 128/4 * 2 + len(''ip6.arpa.'')$;
Function enumerate(base, records={ }, max.ip6.arpa.len)
    for i in 0..f do
        $newbase \leftarrow$ i+"."+base;
        $qryresult \leftarrow$ getptr(newbase);
        if qryresult != NXDOMAIN then
            if len(newbase) == max.ip6.arpa.len then
                add(records, newbase);
            else
                enumerate(newbase,records,max.ip6.arpa.len);
```

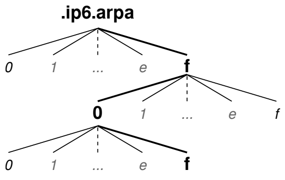


Fig. 1. Enumerating **f.0.f.-ip6.arpa.**, existing nodes in bold.

Specifically: Starting at the root (or any other known subtree), a request for each of the possible child nodes is performed. If the authoritative server returns NXDOMAIN, the entire possible subtree can be ignored, as it indicates that no entries below the queried node exist. Algorithm 1 shows the corresponding algorithmic description. Figure 1 provides a simplified visualization, e.g., if a queries for *0-e.ip6.arpa.* return NXDOMAIN, but *f.ip6.arpa.* returns NOERROR, we can ignore these subtrees, and continue at *f.ip6.arpa.*, finally finding *f.0.f.ip6.arpa.* as the only existing record.

4 Methodology and Algorithmic Implementation

The approach outlined in Section 3 has been used on small scales in the past: Foremski et al. [8] used it to collect a sample of 30,000 records from selected networks for their study. In this section, we analyze the challenges of a global application of the technique and describe how we can overcome these limitations.

Non RFC8020-compliant Systems: The current technique requires that RFC8020 [**?**] is correctly implemented, i.e., that the nameserver behaves standard-compliant. However, following RFC7707 [10], this is not the case for all authoritative DNS nameserver software found in the wild [**?**]. Specifically, if higher level servers (from a DNS tree point of view) are not enumerable by any of the presented techniques, then this can mask the enumerable zones below them. For example, if a regional network registry, like APNIC or, RIPE would use a DNS server that cannot be exploited to enumerate the zone, then all networks for which they delegate the reverse zones would become *invisible* to our methodology.

To approach this challenge, we seed the algorithm with potentially valid bases, i.e., known to exist *ip6.arpa.* zones. Our implementation obtains the most recent Routeviews [19], and the latest RIPE Routing Information Service (RIS) [17] Border Gateway Protocol (BGP) tables as a source. Particularly important to allow the approach to be easily reproducible: both are public BGP view datasets, available to any researcher.

Based on the data, we create a collapsed list of prefixes. Following prior work, we consider the generated list a valid view on the Global Routing Table (GRT) [21]. For each of the collapsed prefixes we calculate the corresponding ip6.arpa. DNS record. The resulting list is then used as the input seed for our algorithm. Alternative public seed datasets are the Alexa Top 1,000,000 [?,5] or traceroute datasets [8] (which, as aforementioned, are biased by nature; thus, special care must be taken for traceroute datasets). If available, other non-public datasets like the Farsight DNS recursor dataset [20] could also be used.

Complimentary approaches to collect ip6.arpa. addresses or subtrees from systems that implement RFC8020 incorrectly are those with which one can obtain (part of) a DNS zone. For example, by employing insufficiently protected domain transfers (AXFRs), which are a prominent misconfiguration of authoritative nameservers [1].

Breadth-First vs. Depth-First Enumeration: For our data collection, we employ Algorithm 1. Unfortunately, the algorithm leverages depth-first search to explore the ip6.arpa. tree. This search strategy becomes problematic if any of the earlier subtrees is either rather full (non-sparse) or if the authoritative nameservers are relatively slow to respond to our queries. Slow responses are particularly problematic: they allow an "early" subtree to delay the address collection process significantly.

Substituting depth-first search with breadth-first search is non-trivial unfortunately. Therefore, we integrate features of breadth-first search into the depth-first algorithm (Algorithm 1), which requires a multi-step approach: Starting from the seed set, we first use Algorithm 1 to enumerate valid ip6.arpa. zones below the records up to a corresponding prefix-length of 32 bits. If we encounter input-records that are more specific than 32 bits, we add the input record and the input record's 32-bit prefix to the result set. Once this step has completed for all input records, we conduct the same process on the result set, but with a maximum prefix-length of 48 bits, followed by one more iteration for 64-bit prefixes. We opted to use 64 bits as the smallest aggregation step because it is the commonly suggested smallest allocation size and designated network size for user networks [11]. Algorithm 2 provides a brief description of the cook_down algorithm. The last step uses Algorithm 1 on these /64 networks with a target prefix size of 128 bits, effectively enumerating full ip6.arpa. zones up to their leaf nodes. To not overload a single authoritative server, the ip6.arpa. record sets are sorted by the least significant nibble of the corresponding IPv6 address first before they are further enumerated. Sorting them by the least significant nibble spreads zones with the same most significant nibbles as broadly as possible.

Combined with the observed low overall traffic that our modified technique generates, we can prevent generating unreasonably high load on single authoritative nameserver. Our approach, contrary to prior work, does not generate high load on the authoritative nameservers before moving on to the next one. Otherwise it would launch a denial of service attack against the nameserver. If our approach is more widely adopted by researchers, future work should investigate how distributed load patterns can be prevented, i.e., thousands of researchers querying the same nameserver simultaneously (see Section 4).

Detecting Dynamically-generated Zones: Dynamically generating the reverse IP address zone, i.e., creating a PTR record just-in-time when it is requested, has been popular in the IPv4 world for some time [15]. Unsurprisingly, utilizing dynamically generated IPv6 reverse zones has become even more common over time as well. Especially

Algorithm 2: Algorithm cooking down the initial seed records.

```
Function cook_down (records)
    for prefix.len in 32,48,64 do
        $records.new \leftarrow \{ \}$;
        $cur.ip6.arpa.len \leftarrow prefix.len/4 * 2 + len(''ip6.arpa.'')$;
        for base in records do
            // See Section 4 Dynamically-generated Zones/Prefix Exclusion/Opt-Out for details;
            if checks(base) == False then
                pass
            else if len(base) $\geq$ cur.ip6.arpa.len then
                $add(records.new, base)$;
                $crop.base$ = croptolength(base,cur.ip6.arpa.len);
                $add(records.new, crop.base)$;
            else
                $add(records.new, enumerate(base, cur.ip6.arpa.len))$;
```

access networks tend to utilize dynamically-generated reverse records. While this provides a significant ease-of-use to the network operators, our algorithm will try to fully enumerate the respective subtrees. For a single dynamically-generated /64 network it leads to 2^{64} records to explore, which is clearly impractical. Therefore, we introduce a heuristic to detect if a zone is dynamically-generated, so that we can take appropriate action. To detect dynamically-generated reverse zones, we can rely on the semantic properties of reverse zones. The first heuristic that we use is the repeatability of returned FQDNs. Techniques for dynamically-generated reverse zones usually aim at providing either the same or similar fully-qualified domain names (FQDNs) for the reverse PTR records. For the former detection is trivial. In the latter case, one often finds the IPv6 address encoded in the returned FQDN. In turn, two or more subsequent records in an dynamically generated reverse zone file should only differ by a few characters. Therefore, a viable solution to evaluate if a zone is dynamically-generated is the Damerau-Levenshtein distance (DLD) [7].

Unfortunately, we encountered various cases where such a simplistic view is insufficient in practice. For instance, zones may also be dynamically-generated to facilitate covert channels via DNS tunneling [13]. In that case, the returned FQDNs appear random. Similarly in other cases, the IPv6 address is hashed, and then incorporated into the reverse record. In those cases the change between two records can be as high as the full hash-length of the utilized hash digest. We devised another heuristic based on the assumption that if a zone is dynamically-generated, then all records in the zone should be present. Following prior work by Plonka et al. and Foremski et al. [14, 8], we determined that certain records are unlikely to exist in one zone all together, specifically, all possible terminal records of a base that utilize only one character repeatedly. For example, for the base *0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa* such a record would be *f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa*. Therefore, we build and query all sixteen possible records from the character set 0..f. Due to these records being highly unlikely [8], and the use of packet-loss sensitive UDP throughout DNS, we require only three records to resolve within a one second timeout to classify a zone as dynamically-generated. We omit the heuristic's algorithmic description for brevity, as the implementation is straight forward.

Algorithm 3: Call-order in final script.

```
$seeds \leftarrow$ get_seeds();
$enum.records \leftarrow$ cook_down(seeds);
$final.result \leftarrow \{\ \}$;
for base in enum.records do
    // See Section 4 Dynamically-generated Zones/Prefix Exclusion/Opt-Out for details;
    if checks(base) $==$ False then
        return $\{\ \}$ ;
    $tmp.results \leftarrow$ enumerate(base, 128);
    $final.result \leftarrow final.result + tmp.results$;
```

Prefix Exclusion: Naturally, in addition to excluding dynamically-generated zones, a network operator may ask to be excluded from her networks being scanned. During our evaluation, multiple network operators requested being excluded from our scans. Furthermore, we blacklisted two network operators that did use dynamically-generated zones, but for which our heuristic did not trigger, either due to rate-limiting of our requests on their side, or bad connectivity toward their infrastructure. Similarly, our algorithm missed a case for a US based university which used /96 network access allocations, which we did not detect as dynamically-generated due to the preselected step-sizes for Algorithm 2. In total, we blacklisted five ISPs' networks and one university network.

Ethical Considerations and Opt-Out Standard: To encourage best practice, for our experiments and evaluation, the outbound throughput was always limited to a maximum of 10 MBit/s in total and specifically to 2MBit/s for any single target system at a time following our least-significant byte sorting for ip6.arpa zones. Although the load we incurred was negligible for the vast majority of authoritative nameservers, we acknowledge that the load this methodology may put onto authoritative servers may become severe, particularly if more researchers utilize the same approach simultaneously or do not limit their outbound throughput. Hence, we suggest to adopt and communicate the practice of first checking for the existence of a PTR record in the form of *4.4.4.f.4.e-.5.4.5.3.4.3.4.1.4.e. ... .ip6.arpa.*. The respective IPv6 record encodes the ASCII representation of DONTSCAN for /64 networks. For networks larger than /64, we suggest to repeat the string. We do not use a non-PTR conform record, as this would exclude users utilizing, e.g., restrictive DNS zone administration software possibly sanitizing input. We will carry this proposal toward the relevant industry bodies, to provide operators a simple method to opt out of scans.

CNAMEs: Our investigation also found cases of seemingly empty terminals in the DNS tree, i.e., records of 32 nibble length without an associated PTR resource record that do not return NXDOMAIN. Upon removal of these records, and by focusing on non-empty terminals in these address bases, we still obtain valid results. In addition to cases where the terminals are fully empty, CNAME records [12] may exist instead of PTR records, which is why it is necessary to resolve CNAME records if a PTR record does not exist.

Parallelization: Combining the previously presented algorithms, we can enumerate the IPv6 PTR space (see Algorithm 3). Due to our algorithm's nature, parallelization is ideally introduced in the *for* loop starting at line 5 of Algorithm 2 and the *for* loop at line 4 in Algorithm 3. Technically, it would also be possible to introduce parallelization in the

Experiment	Runtime					Records Found				Addresses		Queries	Dynamic Zones			Blacklisted	
	/32	/48	/64	Full	Total	Seed	/32	/48	/64	Total	Unique		/32	/48	/64	/32	/48
ip6.arpa.	120	130	429	3,244	3,932	/	3.5k	52.5k	1M	1.6M	335k	62M	615	15k	223k	0	1.5k
GRT_SEED$_{80}$	7	232	1,040	2,956	4,235	72k	73k	856k	582k	5.3M	2.8M	221.3M	1.5k	716k	80.5k	713	63
GRT_SEED$_{400}$	7	144	404	775	1,330	72k	73k	834k	1.4M	2.2M	33k	190.7M	1.5k	690k	796k	715	65
Unique Sum						73k	75k	895k	2,2M	5.8M			1.5k	732k	1M	715	1.6k

Table 1. Overview of the results of our evaluation.

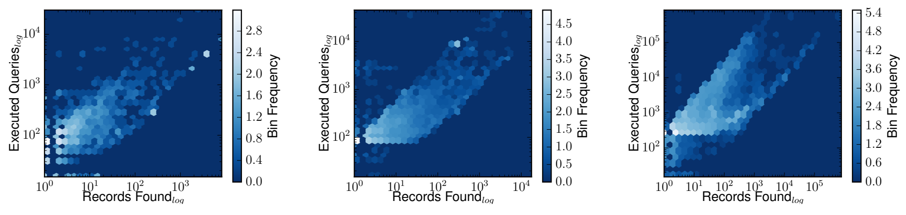
first *for* loop of Algorithm 1. However, then parallelization might be performed over a single authoritative server. This would put a high load on that system. By parallelizing our approach through Algorithm 2 and Algorithm 3 parallel queries are made for different IPv6 networks, thus most likely to different authoritative servers.

5 Evaluation

We evaluate our methodology on a single machine running Scientific Linux 6.7 with the following hardware specification: four Intel Xeon E7-4870 CPUs (2.4GHz each) for a total of 80 logical cores, 512GB of main memory, and 2TB of hard-disk capacity. We installed a local recursive DNS resolver (Unbound 1.5.1) against which we perform all DNS queries. Connection-tracking has been disabled for all DNS related packets on this machine, as well as other upstream-routers for DNS traffic from this machine. An overview of our results can be found in Table 1.

Enumerating .ip6.arpa.: In our first evaluation scenario, we enumerate addresses using the PTR zone root node of .ip6.arpa. as the initial input only, which will serve as basic ground-truth. The respective dataset corresponds to the first column of Table 1: ip6.arpa. The enumeration was completed within 65.6 hours, of which most time was spent enumerating pre-identified /64s networks. As such, the impact of dynamic-generation is evident from this experiment: 615 /32 prefixes are ignored due to dynamically-generated PTR records, with an additional 15k /48 prefixes and more than 223k /64 networks subsequently. This experiment yields a total of 1.6 million allocated IPv6 addresses.

GRT_SEED$_{80}$: Seeded Enumeration (80 Threads): For our second experiment, we used the current IPv6 GRT as a seed and ran our algorithm with 80 threads in parallel. The respective dataset is identified as GRT_SEED$_{80}$ in Table 1. The GRT is compiled following our description in Section 4. In contrast to simply enumerating the ip6.arpa. zone, pre-aggregating to /32 prefixes takes significantly less time. The reduced time is primarily due to the seeds in the GRT having a certain prefix length already, mostly /32 prefixes. The same can be observed when comparing the seed set among aggregated /32 prefixes. Interestingly, the dataset only increases by around 1,000 prefixes in that aggregation step, mostly due to longer prefixes being cropped. However, in the next step, we do find a significantly larger number of prefixes than those contained in the seed set. Unfortunately, the next aggregation step demonstrates that a significant amount of them are in fact dynamically-generated client allocations. Nonetheless, at more than 5.4 million unique allocated IPv6 address collected, leveraging the GRT seed to improve collection exceeds the initial dataset by far (1.6 milion to 5.4 million). It is important



(a) Enum. to /48 (b) Enum. to /64 (c) Enum. to /128

Fig. 2. Executed DNS queries vs. obtained records for GRT_SEED$_{80}$.

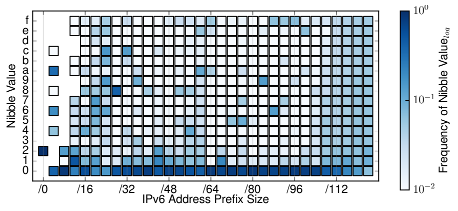
to note, however, that we discovered 335,670 records that are unique to the ip6.arpa. dataset. These originate from currently unannounced prefixes. The ip6.arpa. root-node should hence be included into every seed-set. However, depending on the purpose of the data collection, identified yet unrouted addresses should be marked in the collected data set.

GRT_SEED$_{400}$: Seeded Enumeration (400 Threads): Unfortunately, a full run with 80 parallel threads takes nearly three full days to complete. Therefore, a higher time resolution is desirable. Due to low CPU load on the measurement machine we investigated the impact of running at a high parallelization degree, using 400 threads to exploit parallelization more while waiting for input/output. We refer to this dataset as GRT_SEED$_{400}$, which was collected in less than a day. In comparison to collecting with less parallel threads, we do not see a significant impact at the first aggregation level toward /32s prefixes (which we expected) due to the generally low number of them that must be enumerated here.

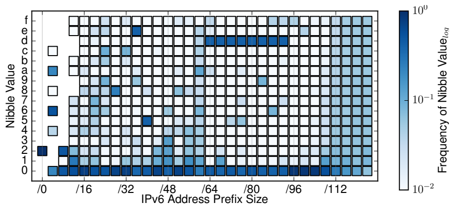
At the same time, we see a far higher number of obtained prefixes, primarily /64 prefixes. However, when examining the number of detected dynamically-generated and blacklisted prefixes closer, we do see that a number of dynamically-generated prefixes is not being detected correctly, which we discovered is due to packet loss. This is highlighted by the number of prefixes in GRT_SEED$_{400}$ for each aggregation level, which are considered dynamically-generated in a less specific aggregation level of GRT_SEED$_{80}$. Indeed, for 92.94% of dynamically-generated /64 in GRT_SEED$_{400}$, they have a /48 prefix already considered dynamically-generated in GRT_SEED$_{80}$.

Although the results between GRT_SEED$_{80}$ and GRT_SEED$_{400}$ differ significantly, CPU utilization for GRT_SEED$_{400}$ was not significantly higher. The core reason for this behavior is that our technique is not CPU bound. Instead, the number of maximum sockets and in-system latency during packet handling have a significantly higher impact on the result. Hence, instead of running the experiment on a single host, researchers should opt to parallelize our technique over multiple hosts.

Queries per Zone and Records Found: The number of queries sent to each /32, /48 and /64 prefixes respectively versus the number of more specific ip6.arpa. records obtained per input prefix is contrasted in Figure 2(a)-2(c). An interesting insight of our evaluation is that most zones at each aggregation level contain only a limited set of records. Furthermore, we discover that the number of records found versus the number of executed queries is most densely populated in the area of less than 10 records



(a) Combined Result Set



(b) Biased Data Acquisition

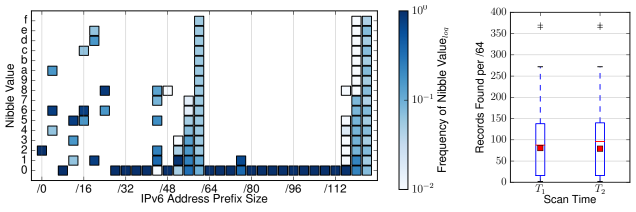
Fig. 3. Probability mass function for each 4bit position in obtained datasets following Foremski et al. [8]. Figure 3(a) visualizes our combined dataset, with 5,766,133 unique IPv6 addresses. Figure 3(b) depicts an artifact from a measurement error in an earlier study.

per zone. Additionally, we see a clear lower-bound for the number of required queries. Specifically, the lower bound consists of the 16 queries needed to establish if a zone is dynamically-generated, plus the minimum number of queries necessary to find a single record. Correspondingly, for the de-aggregation to /64, an additional 64 queries are required. To go from an aggregation level of /64 to a single terminal record, at least 256 queries are necessary.

Clear upper and lower bounds for the quotient of executed queries and obtained records are also visible. In fact, these bound become increasingly clear while the aggregation level becomes more specific and follows an exponential pattern, hinting at an overall underlying heavy-tailed distribution. Furthermore, the two extremes appear to accumulate data-points, which is evident from Figure 2(c). The upper bound thereby corresponds to zones with very distributed entries, i.e., zones that require a lot of different paths in the PTR tree to be explored, e.g., zones auto-populating via configuration management that adds records for hosts with stateless address auto-configuration (SLAAC). On the other hand, the lower bound relates to well-structured zones, i.e., for which the operators assign addresses in an easily enumerable way, e.g., sequentially starting at *PREFIX::1*.

Address Allocation: We utilized the visualization technique introduced by Foremski et al. [8] to analyze our dataset. To do so, we created the set of all unique IPv6 address records we obtained over all measurements. The respective results are depicted in Figure 3: the least significant nibbles are relatively evenly distributed, which aligns with our observation that zones are either very random or in some form sequential.

Fortunately, the technique by Foremski et al. [8] also allows us to validate our dataset. Specifically, Figure 3(b) has been created over an earlier dataset that we collected where an unexpected summation of the value *d* in IPv6 addresses between the 64_{th} and 96_{th} bit appears. A closer investigation revealed that this artifact was caused by a US-based educational institution that uses their *PREFIX:dddd:dddd::/96* allocation for their DHCPv6 Wi-Fi access networks. As aforementioned, this dynamically-generated network was not detected due to the step-sizes in Algorithm 2, which is why we excluded it manually, see Section 4. Further work should consider adopting, e.g., 4 nibble wide steps.



(a) Density in SaaS provider at T_2 (b) Addr. per /64

Fig. 4. Overview of address allocation in the SaaS cloud provider's network.

6 Case-Study

Following, we present how findings of our technique can be used to obtain in-depth insights into practical issues. We provide a brief analysis of the IPv6 efforts in the internal infrastructure of a large SaaS (Software-as-a-Service) cloud platform operator. For our investigation, we selected the prefixes of this operator based on its IPv6 announcements collected via bgp.he.net. To obtain further ground-truth, we also collected the PTR records for all IPv4 prefixes announced by the operator's autonomous system (AS) from bgp.he.net. We took two measurements, T_1 and T_2, two weeks apart in September 2016. Figure 4 shows an overview of the allocation policy of the operator. Specifically, the operator uses three /32 prefixes, with one being used per region she operates in (see Figure 4(a)). In each region, the operator splits her prefix via the 40_{th} to 44_{th} bit of addresses. IPv6 networks used by network-edge equipment for interconnectivity links between different regions are distinguished by an *8* at the 48_{th} to 51_{st} bit, instead of *0*, which is used by all other prefixes.

Another interesting part of the addressing policy are the /48 networks the SaaS provider allocates. Here, we can see that networks are linearly assigned, starting with *PREFIX:0000::/48*, thus creating pools of /64s for various purposes. Furthermore, with /48s being linearly assigned, we discover that prefixes with higher indexes have not yet been assigned. The same assignment policy holds for hosts in /64s networks, as indicated by the distribution over the three least significant nibbles used in addresses.

A third aspect of the operator's assignment policy is documented in Figure 4(b). Specifically, the boxplots show the number of hosts per /64 prefix in the operators networks. For both measurements, we only observe two /64 prefixes with significantly more than 250 hosts. A closer investigation of these networks reveals that they are related to internal backbone and firewalling services spanning multiple Points-of-Presence, following the PTR naming schemes of the obtained records. Apart from this change, we do see a slight increase in the number of hosts per network in the median, but not the mean. An interesting side-note is that the IPv6 PTR records appear manually allocated by the operator's network staff. We do arrive at this conclusion because we encountered various records with typographical errors in them.

Comparing of the datasets with the corresponding IPv4 PTR sets, we note that the diversity of records is far higher in the IPv4 set. There, various second-level domains can be found mixed together, which we did not encounter for the IPv6 set. Various naming schemes for infrastructure hosts are also present. For example, we discover that the customer-facing domain of the operator is being used for infrastructure services. However, it has apparently been disbanded with the growth of the organization, as we also discover infrastructure specific second-level domains. For the IPv6 set we only observe one infrastructure domain. In general, naming is far more consistent for IPv6. Our conjecture is that the operator made an effort in keeping a consistent state when finally rolling out IPv6, while IPv4 is suffering from legacy setups introduced during the company's growth. The last striking observation is that the PTR records returned for IPv4 and IPv6 reverse pointers do not resolve to valid A and AAAA records themselves. A direct consequence is that, for this network operator, the technique proposed by Czyz et al. [5] is not applicable. We conjecture that the operator chose this setup because she does not require forward lookups, yet wants traceroutes and other reverse-lookup related tools, especially distributed logging, to show the FQDNs.

7 Conclusion

We introduce a novel methodology to collect a large IPv6 dataset from exclusively public data sources. Our initial evaluation of the methodology demonstrates its practical applicability. Requiring no access to a specific network vantage point, we were able to collect more than 5.8 million allocated IPv6 addresses, of which 5.4 million addresses were found in just three days by issuing 221 million DNS queries. Specifically, our technique discovered one allocated IPv6 address per only 41 DNS queries on average. With the obtained dataset, we were able to provide an in-depth look into the data-centers of a large cloud provider. By comparing our results with the corresponding IPv4 reverse entries, we demonstrate that our technique can discover systems which would have been missed by previous proposals for collecting IPv6 addresses [5]. In summary, our technique is an important tool for tracking the ongoing deployment of IPv6 on the Internet. We provide our toolchain to researchers as free software at: *https://gitlab.inet.tu-berlin.de/ptr6scan/toolchain*

We note that our technique can also be applied to E.164 records (Telephone Numbers in DNS), but leave this for future work. Furthermore, future work should utilize this technique over a period of time in order to obtain a progressing view on IPv6 deployment on the Internet. To increase coverage, additional seeds and other address collection techniques should be integrated. This should be combined with security scanning as it is common practice for IPv4 [18]. Following the findings of Czyz et al. [5], such projects are direly needed to increase overall security on the Internet.

Acknowledgements. We thank the anonymous reviewers for their helpful feedback and suggestions, and Peter van Dijk for suggesting this research path to us. This material is based on research supported or sponsored by the Office of Naval Research (ONR) under Award No. N00014-15-1-2948, the Space and Naval Warfare Systems Command (SPAWAR) under Award No. N66001-13-2-4039, the National Science Foundation (NSF) under Award No. CNS-1408632, the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8750-15-2-0084, a Security, Privacy and Anti-Abuse award from Google, SBA Research, the Bundesministerium für Bildung und Forschung (BMBF) under Award No. KIS1DSD032 (Project Enzevalos), a Leibniz

Price project by the German Research Foundation (DFG) under Award No. FKZ FE 570/4-1. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The opinions, views, and conclusions contained herein are those of the author(s) and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ONR, SPAWAR, NSF, DARPA, the U.S. Government, Google, SBA Research, BMBF, or DFG.

References

1. Atkins, D., Austein, R.: Threat Analysis of the Domain Name System (DNS). RFC3833 (2004)
2. Bortzmeyer, S., Huque, S.: NXDOMAIN really means there is nothing underneath. draft-ietf-dnsop-nxdomain-cut-05 (2016)
3. Chatzis, N., Smaragdakis, G., Böttger, J., Krenc, T., Feldmann, A.: On the benefits of using a large ixp as an internet vantage point. In: Proc. ACM Internet Measurement Conference. pp. 333–346 (2013)
4. Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., Bailey, M.: Measuring IPv6 Adoption. Proc. ACM SIGCOMM 44(4), 87–98 (2015)
5. Czyz, J., Luckie, M., Allman, M., Bailey, M.: Don't forget to lock the back door! a characterization of ipv6 network security policy. In: Proc. Symposium on Network and Distributed System Security (NDSS). vol. 389 (2016)
6. Durumeric, Z., Wustrow, E., Halderman, J.A.: Zmap: Fast internet-wide scanning and its security applications. In: Proc. Usenix Security Symp. pp. 605–620 (2013)
7. Fiebig, T., Danisevskis, J., Piekarska, M.: A metric for the evaluation and comparison of keylogger performance. In: Proc. USENIX Security Workshop on Cyber Security Experimentation and Test (CSET) (2014)
8. Foremski, P., Plonka, D., Berger, A.: Entropy/ip: Uncovering structure in ipv6 addresses. In: Proc. ACM Internet Measurement Conference (2016)
9. Gasser, O., Scheitle, Q., Gebhard, S., Carle, G.: Scanning the ipv6 internet: Towards a comprehensive hitlist (2016)
10. Gont, F., Chown, T.: Network Reconnaissance in IPv6 Networks. RFC7707 (2016)
11. Hinden, R., Deering, S.: IP Version 6 Addressing Architecture. RFC4291 (2006)
12. Mockapetris, P.: Domain names - implementation and specification. RFC1035 (1987)
13. Nussbaum, L., Neyron, P., Richard, O.: On robust covert channels inside dns. In: Proc. International Information Security Conference (IFIP). pp. 51–62 (2009)
14. Plonka, D., Berger, A.: Temporal and spatial classification of active ipv6 addresses. In: Proc. ACM Internet Measurement Conference. pp. 509–522. ACM (2015)
15. Richter, P., Smaragdakis, G., Plonka, D., Berger, A.: Beyond Counting: New Perspectives on the Active IPv4 Address Space. In: Proc. ACM Internet Measurement Conference (2016)
16. Ripe, NCC: RIPE atlas, `http://atlas.ripe.net`
17. RIPE NCC: RIPE Routing Information Service (RIS), https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
18. ShadowServer Foundation: The scannings will continue until the internet improves. http://blog.shadowserver.org/2014/03/28/the-scannings-will-continue-until-the-internet-improves/ (2014)
19. University of Oregon: Route Views Project , `http://bgplay.routeviews.org`

20. Vixie, P.A.: It's time for an internet-wide recommitment to measurement: And here's how we should do it. In: Proc. International on Workshop on Traffic Measurements for Cybersecurity. pp. 1–2 (2016)
21. Zhang, B., Liu, R., Massey, D., Zhang, L.: Collecting the internet as-level topology. ACM Computer Communication Review 35(1), 53–61 (2005)