Glowing in the Dark
Uncovering IPv6 Address Discovery and Scanning Strategies in the Wild

Hammas Bin Tanveer[1], Rachee Singh[2,3], Paul Pearce[4], Rishab Nithyanand[1]
[1]University of Iowa, [2]Microsoft, [3]Cornell University, [4]Georgia Tech

Abstract

In this work we identify scanning strategies of IPv6 scanners on the Internet. We offer a unique perspective on the behavior of IPv6 scanners by conducting controlled experiments leveraging a large and unused `/56` IPv6 subnet. We selectively make parts of the subnet visible to scanners by hosting applications that make direct or indirect contact with IPv6-capable servers on the Internet. By careful experiment design, we mitigate the effects of hidden variables on scans sent to our `/56` subnet and establish causal relationships between IPv6 host activity types and the scanner attention they evoke. We show that IPv6 host activities e.g., Web browsing, membership in the NTP pool and Tor network, cause scanners to send a magnitude higher number of unsolicited IP scans and reverse DNS queries to our subnet than before. DNS scanners focus their scans in narrow regions of the address space where our applications are hosted whereas IP scanners broadly scan the entire subnet. Even after the host activity from our subnet subsides, we observe persistent residual scanning to portions of the address space that previously hosted applications.

1 Introduction

Scanning the IP address space has exposed security vulnerabilities, enabling researchers and practitioners to develop effective defenses. Tools for scanning the IP address space grapple with the fundamental challenge of efficient scanner target discovery — a task made more challenging by the increasing adoption of IPv6 on the Internet. The IPv6 address space consists of 2^{128} possible addresses, rendering brute-force generation of scanning targets infeasible. Recent work has developed tools [9, 17] to make Internet-scale IPv6 scanning practical by analyzing IPv6 address assignment patterns [40, 53, 54] and developing efficient scanner target generation algorithms [40, 53, 54].

Despite the recent work on effectively scanning the IPv6 address space, little is known about the scanning strategies deployed in the wild. We address this gap by analyzing IPv6 scanning from the perspective of IPv6 hosts on the Internet. Our goal is to reveal target generation strategies of IPv6 scanners and inform address assignment policies to mitigate the impact of Internet-scale IPv6 scanners. Previous work with similar goals performed observational studies using passive measurements of unsolicited traffic. While observational studies provide useful insights they (1) do not identify address discovery strategies leveraged by scanners and (2) may not be representative of the real scanning activity observed by actively in-use IPv6 networks [26, 34, 36, 46] (§2). In contrast, we take an active approach by conducting *controlled experiments* to evaluate the impact of IPv6 host activity on scanner behavior. We begin the study by acquiring a previously-unused `/56` IPv6 subnet owned by a university. This address space did not originate any traffic prior to the start of the study, allowing us to conduct clean-slate controlled experiments that make parts of the address space visible on the Internet for the first time during our study. A subgroup of *treatment* subnets in our address space host applications that make direct or indirect contact with potential IPv6 scanners. We measure the *effect* of the treatment by capturing unsolicited IPv6 scanning activity received by our address space. By comparing the impact on scanning activity between the treatment and *control* subnets, we establish causal relationships between types of host activities and increased scanner attention (§3).

Accurately associating a measurable increase in scanner attention to specific IPv6 host activities is challenging. First, due to our large yet limited IPv6 address space for experimentation, discerning the effect of one host activity (Web browsing) from another (Tor relay) on increased scanner attention is hard as the observable scanning activity can result from a combination of host activities. Second, scanning activity can often persist after the experimental host activity has subsided, confounding the measured effects of subsequent experiments. Finally, Internet scanners can coincidentally scan our treatment subnets, endangering false conclusions about the effect of host activity on scanner attention. We carefully design our controlled experiments to mitigate the effects of these hidden variables to improve the accuracy of our conclusions (§4). Our study spans over a year from the time we acquired the `/56` subnet and began the controlled experiments. We analyze the IPv6 scans and reverse DNS queries destined for our `/56` address space to make the following key findings:

Host activity has a sizable impact on scanner attention. While our subnet attracted moderate background radiation scans before we began our controlled experiments to simulate host activity, the host activity evoked sizable scanner attention — with IP and DNS scanning increasing to 225$\times$ and 1.6$\times$ pre-experiment rates, respectively. On conclusion of our experiments, the rates continued to stay high at 426$\times$ and 3.7$\times$, respectively. Moreover, host activity that makes direct contact with IPv6 capable servers on the Internet (e.g., Web browsing, querying open DNS resolvers) evokes 50$\times$

and 13× more activity during and after experimentation, in comparison with host activities which make indirect contact with potential scanners (§3).

DNS scanners have a narrow attention space. Reverse-DNS scanners focus their attention on scanning the treatment subnets i.e., subnets hosting the IPv6 application and send very few scans to subnets outside the treatment subnet. In contrast, IP scanners have a broader attention span, often scanning both inside and outside the treatment subnets (§4).

Residual scanning after host activity subsides. DNS scanning activity persists long after the host activity that evoked the scans has subsided. In specific, we observe high volume DNS scans for nearly six weeks in the treatment subnets used for browsing the Web after the Web browsing activity has ended. The scans re-start after a break of 2-3 weeks (§4).

Random and low-byte scanning are dominant strategies. We fingerprint the IPv6 scanners that sent unsolicited traffic to our subnet during the study and find that IPv6 scanners have one of two main scanning strategies: they either have equal interest in the entire address space (random scanners) or focus more on low-byte addresses (§5).

Finally, we discuss the implications of our findings for network operators (§6) and place our contributions in the context of previous work (§7).

2 Background

In this section, we provide a high-level overview of IPv6 addressing and IPv6 scanning strategies.

IPv6 addressing. An IPv6 address consists of 128 bits which may be broken down into three parts — an m bit routing prefix, an n bit subnet ID, and a k (= 128-m-n) bit interface ID (IID). The routing prefix and subnet ID are used to route traffic to a local network where hosts are identified by unique IIDs [44]. Network operators face two questions while allocating IPv6 addresses to networks and hosts (1) how many bits of an IPv6 address should be allocated to host IIDs (i.e., the value of k) and (2) how should IIDs be allocated to hosts in a subnet.

Determining IID lengths. It is considered the best practice for network operators to leave 64 bits for IIDs according the RFC4291 [44]. There are many compelling reasons for this practice. First, a large number of existing IPv6 configuration options and RFC recommendations assume a 64-bit IID. Therefore, not following this recommendation may result in operational failure when using IPv6-specific features and technologies. For example, as pointed out in RFC 4291, a 64-bit IID is required to use privacy-enhancing IPv6 features which allow for cryptographically generated addresses (RFC 3972 [23]), or to use IPv4-to-IPv6 transition protocols such as 6to4 (RFC 3056 [52]), or to leverage neighbor discovery protocols implemented in accordance with RFC 4861 [51].

Assigning IIDs. IIDs may be allocated to hosts through (1) manual configuration, (2) stateless address auto configuration (SLAAC defined in RFC4862) [58], or (3) the DHCPv6 IP leasing protocol (defined in RFC 8415 [49,55]). Each of these three methods may result in host addresses having different IID characteristics. For example, network operators using manual configurations may assign host IPs in a predictable manner — e.g., using sequential addressing which results in the lower bytes of the IID being populated with all leading bytes set to 0 (as per RFC7707 [40]) or assign host IP addresses based on their 48-bit MAC addresses (the modified EUI-64 protocol defined in RFC4291 [44]). On the other hand, operators using SLAAC or other approaches (e.g., from RFC 7943 [41], RFC 7217 [39], and RFC 3972 [23]) generate pseudo-random IIDs.

Our address allocation approach. Given the importance of a 64-bit IID length, it is reasonable to assume that scanners assume 64-bit IIDs in their scanning targets. Therefore, in all our subsequent experiments detailed in §3 and §4, we assign 64-bit IIDs to all our hosts and subnets. Further, we use both IID generation approaches (lower-byte and pseudo-random addresses generation) to generate IPs for hosts whose addresses are leaked to scanners. This allows us to study the impact of address allocation methods on the subsequent address generation strategies leveraged by IPv6 scanners — e.g., does the discovery of a host with a pseudo-random (or, lower-byte) IID result in scanners only sending probes to other addresses with a pseudo-random (or, lower-byte) IIDs.

IPv6 address representation. A common method for representing IPv6 addresses uses 32 *nybbles*, each denoted in hexadecimal and representing four contiguous bits. Every four nybbles (two bytes) are separated by a ':' and any leading zeros in these two byte sections may be dropped.

Representation in DNS reverse zones. DNS reverse zones map addresses to domains — i.e., the opposite of what DNS zones do. Sending a DNS PTR query to the corresponding IPv6 reverse zone returns the domains hosted with the corresponding IP address. IPv6 PTR records are organized under `ip6.arpa` in 32 levels where each nybble of an exploded IPv6 address is a level. Therefore, the PTR record associated with the address `2601::dead:1` is available at `1.0.0.0.d.e.a.d.<20 repeating .0s>.1.0.6.2.ip6.arpa`.

Scanning strategies: address discovery. Unlike IPv4, scanning the entire IPv6 address space for live addresses is infeasible. Instead, scanners focus their attention on regions of the address space near addresses where some activity has been observed. Prior work has used techniques like monitoring public lists of IPv6 addresses (e.g., TLD zone files which list the IPv6 addresses of domains) or hosting public services to receive contact from previously unseen addresses (e.g., hosting a web service).

Our address leaking strategies. In our controlled experiments described in §3 and §4, we leak the 'liveness' of specific regions of an IPv6 network by the following means: (1) direct contact with IPv6 capable web services, (2) sending DNS queries to public DNS resolvers, (3) participation in the NTP pool protocol [11] where IP addresses of participants can be enumerated [19], (4) participation in the NTP public server protocol [18] where our addresses are distributed via a public listing of NTP servers, (5) participation in the Tor network as a middle-relay where our addresses are distributed via the Tor consensus [13], and (6) registering domains with specific addresses so their presence is known via the TLD zone files.

Scanning strategies: probing for liveness Once scanners have a specific region of the address space to focus on, they may use a different strategy to probe for active hosts.

Traditional IP scanning. A common approach is to solicit responses from live hosts by sending probes (packets associated with a common Internet protocol such as ICMPv6) to a set of candidate IP addresses using tools such as *zmapv6* [3] and *nmap* [7]. These candidates are generated by observing patterns in the already identified live addresses (e.g., known live addresses are sequentially allocated) by off-the-shelf tools like *ipv666* [2] and *the IPv6 toolkit* [10].

NXDOMAIN scanning. An emerging and increasingly popular approach for scanning IPv6 spaces for liveness involves leveraging a feature of the IPv6 reverse-DNS lookup process detailed in RFC8020 [25]. RFC 8020 [25] states that reverse-DNS lookups within subnets that contain no domains should receive an `NXDOMAIN` response code — e.g., if the prefix `2601::dead:1/80` contains no domains, reverse-DNS queries for any more-specific prefixes should return an `NXDOMAIN` response code. This allows scanners to make significant reductions the address search space.

Our data logging approach. In our experiments, we are interested in identifying and detailing the behavior of traditional IP scanners and NXDOMAIN scanners. Therefore, we set up our infrastructure to capture all packets and DNS PTR lookups for our address space.

3 Prevalence of IPv6 Scans

In this section, we answer the question: *How prevalent is IPv6 scanning in the wild?*

3.1 Data collection methodology

Overview. Our study is based on scanner behavior observed on a previously unused and unannounced `/56` IPv6 address space (§3.1.1). We conduct controlled experiments by creating a *treatment group* of `/64` subnets which contain publicly advertised and visible services and a *control group* of subnets with no services (§3.1.2). To log scanning behavior, we use a logging infrastructure that captures all packets and DNS queries sent to our `/56` IPv6 address space (§3.1.3).

3.1.1 Characteristics of our IPv6 address space

Our `/56` subnet is a part of an autonomous system (AS)'s `/48` allocation. Since the `/56` subnet was previously unused and unannounced by BGP, it should not receive any legitimate traffic [16]. We were granted access to the `/56` in *11/2020* after which the parent AS announced it to the Internet via BGP and we setup data logging infrastructure (§3.1.3). We left the address space idle for the next three months to benchmark base levels of Internet background radiation received by our subnet. In *03/2021*, we began a series of controlled experiments to understand IPv6 scanner behavior. Figure 1 summarizes the timeline of our experiments.

Separating our `/56` into treatment and control groups. We run six different controlled experiments on our address space — each requiring four unique end-host IPs. We first subdivided the `/56` address space into four `/58` address spaces. Then, due of the importance of using 64-bit IIDs for each host (*Cf.* §2), we allocated each of our 24 end-hosts to a unique `/64` subnet and assigned them addresses from this subnet. Therefore, each of our six experiments were conducted on four unique end-hosts contained in four unique `/64` subnets. For two of the four end-hosts associated with each experiment, we allocated a pseudo-random IID. The remaining two received lower-byte IIDs. *Thus, we conduct each experiment on two end-hosts which reflect the two most common forms of address assignment in IPv6 networks.* The 24 `/64` subnets associated with our experiments are our *treatment subnets* and every other subnet is a *control subnet*. All treatment subnets were randomly chosen from the same `/58` subnet such that no two were adjacent to each other. This allows for proper analysis of treatment effects (§4) and facilitates retries on the remaining three `/58` subnets if an experiment had to be repeated due to failures. Fortunately, the latter was not required.

3.1.2 Attracting scanner attention

Following the initial three month period of inactivity from 11/2020 to 03/2021, we simulated host activity from our treatment subnets by launching *services* (i.e., experiments[1]) that mimic specific types of host behaviors on the Internet. We ran one service at a time on the four corresponding treatment subnets for *at least* 2 weeks to measure the impact on scanning behavior caused by the *specific host activity* that is mimicked by the service. More details on the methods for measuring service effects are in §4.

Experiments (services) deployed. The goal of our experiments is to identify the effect of IPv6 host activity on scanning behaviors. We achieve this goal by simulating six types of host activity from the 24 `/64` treatment subnets. Each experiment *uses a different method to leak the liveness of the treatment `/64` subnets* to scanners. These methods are based on findings from prior work which highlight the sources of IPv6 addresses leveraged for efficient IPv6 scanning (§7). We leak our addresses using a combination of *direct* and *indirect* scanner contact approaches. Direct contact approaches send packets directly to IPv6 addresses with the expectation of

[1]In the remainder of this paper, experiments and services are used interchangeably.

receiving scanner attention in return. In comparison, indirect contact approaches enlist our services in public lists that may be monitored by scanners seeking to discover new IPv6 addresses. Our six experiment deployments are described below.

Experiment 1: Web crawls to popular websites. With this direct contact experiment, we mimic web browsing from a standard home network where users make connections to web servers that scanners may be operating or tapped into for sources of IPv6 addresses [56]. We first identified all IPv6-capable websites in the Alexa Top 10K websites obtained in 02/2021 by collecting their `AAAA` DNS records and checking them for validity. In total, we found 2.6K IPv6-capable websites which were the subject of our crawls. Following the recommendations of Ahmad et al. [22], we conducted crawls using a simple CLI crawler which did not load third-party or dynamic content (`wget`) and a full-fledged browser using OpenWPM [31]. Our `wget` crawls did not load third-party content and therefore only established direct connections with the web servers of each website while the OpenWPM connections used Firefox to also load dynamic content and make connections with all third-party web servers associated with a website. Therefore, each crawl leaked the same four host addresses to a different (but overlapping) set of web servers. Each crawl was conducted 2 weeks apart.

Experiment 2: Querying DNS open resolvers. In this direct contact experiment, we leak our treatment subnet liveness to open IPv6-capable DNS resolvers. Since no such list of resolvers exists for IPv6, we used the approach of Hendriks et al. [43] to identify IPv6-capable resolvers from IPv4 open resolver lists. In total, we obtained 9K IPv6-capable open resolvers and queried each of them for the `AAAA` record of *www.google.com*. These queries were repeated every day for a two week period. Therefore, this experiment leaked the liveness of four treatment subnets to over 9K IPv6 open resolvers.

Experiment 3: NTP pool servers. In this indirect contact experiment, we hosted four instances of NTP pool servers in four treatment subnets. To ensure that each NTP pool instance used a different egress IP address, we created four network namespaces on our Linux machine to isolate the NTP servers we hosted. Network namespaces ensure separate ports and IP addresses are assigned to them, allowing each of our NTP pool servers to use a different IP address associated with one of the four treatment subnets allocated to this experiment. Each server was initially configured with the NTP default parameters which set rate limits on our responses to liveness probes from other NTP servers for the first two weeks. These rate limits prevented it from achieving the maximum pool score of 20 during this period. During this time, the server was usable by clients (and therefore discoverable by scanners) but not recommended due to a low pool score. In §4 we refer to this part of the experiment as ‘NTP_{pool}’. We removed the rate-limit after two weeks and consequently our servers immediately achieved the maximum pool score of 20 and was recommended for client use. We carried this phase

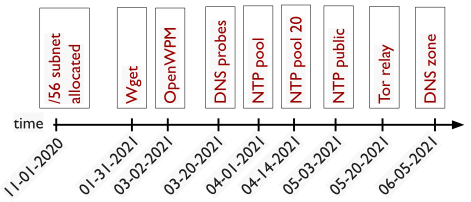


Figure 1: Timeline of our experiments that simulate host activity from a `/56` IPv6 subnet we own.

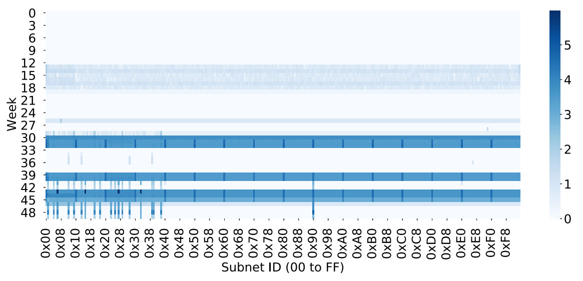
of the experiment for another two weeks and refer to it as ‘$\text{NTP}_{pool-20}$’. Note that NTP pool servers are not publicly listed on a website, but are possible to enumerate [19]. Therefore, this indirect experiment leaked the liveness of its four treatment subnets to scanners that enumerate NTP pool server lists for scanning destinations.

Experiment 4: NTP public servers. While our NTP pool servers were used by clients for synchronizing time, they were not publicly listed and require additional effort to enumerate. In this indirect contact experiment, we launched four NTP *Stratum 2* public servers that remained active for two weeks. Unlike pool servers, these are published on an archived list making them more visible to scanners [12]. This indirect contact experiment leaked the liveness of its four treatment subnets to scanners monitoring NTP server lists.

Experiment 5: Tor relays. For this indirect contact experiment, we launched four Tor [28] middle relays with unrestricted bandwidth in our subnet. These relays remained operational for a two week period. Since our main purpose was to enlist on the public Tor consensus, we chose middle relays as opposed to entry or exit relays. We made this decision because entry relays receive information about the clients connecting to Tor and exit relays receive information about the destinations of Tor traffic. We did not find such information appropriate to gather and analyze. For the additional safety of Tor users, we discarded any non-scanner traffic (defined in §3.1.3) destined for the deployed relays. Therefore, this indirect contact experiment leaked the liveness of the four treatment subnets to any IPv6 scanners monitoring the Tor consensus.

Experiment 6: DNS zone files. Finally, we registered four domains, two each with a `.com` and `.net` TLD. The AAAA records of all four domains pointed to addresses from four of our treatment subnets. Registering these domains with the `com` and `net` TLDs results in them getting added to the largest TLD zone files. Since prior work has leveraged these lists to identify web services with IPv6 addresses, we expect to make indirect contact with scanners monitoring these zone files.

Limitations of our experimental setup. We selected the above-mentioned services for two reasons: (1) they provide a range of common host activities that expose their IP addresses to potential scanners and (2) based on their use in IPv6 target generation and address extraction in prior work [15, 37, 40]. These services allow us to "leak“ our address space to scanners and achieve two goals. First, they help us identify how



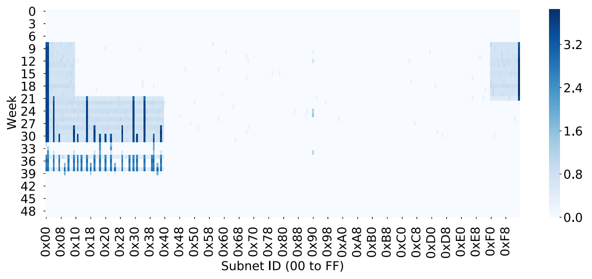


Figure 2: Scanning activity observed by our 256 /64 subnets during each of the 48 weeks in our study. IP scans shown in the top and DNS scans in the bottom figure. Values are $\log_{10}$-scaled.

scanning behaviors can affect commonly used services and second, they provide a method for rigorously measuring the exposure of a specific service to scanners. We note that our choice of deployed services is not comprehensive — i.e., it is possible that other services attract different scanning patterns.

3.1.3 Measuring scanning activity

What is a scanner? Our experiments solicit legitimate traffic amongst traffic from scanners (e.g., NTP clients may send legitimate queries to our NTP pool servers). To differentiate between traffic from legitimate sources and scanners, we analyze the sources of all traffic and label scanners as any sources that send packets to an IP address not associated with any of our experiments. Further, we take a conservative approach and: (1) count all sources emerging from the same /64 subnet as a single scanner — we do this because our data shows several scanners using a distributed infrastructure for scanning (identified by sequential source addresses and wide area scans) and (2) also discount traffic from sources we previously communicated with as part of our experiments. While this results in removing scanners using the same IP address as the service they run (e.g., public DNS resolver), it prevents accidental over counting (e.g., of NTP clients that need to perform reverse-DNS lookups as part of the protocol [12]).

Data logging. To capture IP scanning, we captured PCAPs of all traffic originating from and destined to any address in our /56 allocation. To capture NXDOMAIN scanning, our AS ran an authoritative DNS server which served all domains assigned to our IP address allocation with `TTL = 1`. This low TTL ensured that PTR queries [48] were not cached and continue to reach our authoritative server. We were provided with daily dumps of logs from the server which contained the timestamps, sources, and destinations of queries received.

		DNS logs	PCAPs
# Scanner probes	Before	200,335	13,044
	After	499,638	14,564,017
	Total	699,973	14,577,061
# Scanners	Before	20	5
	After	89	1065
	Total	96	1068

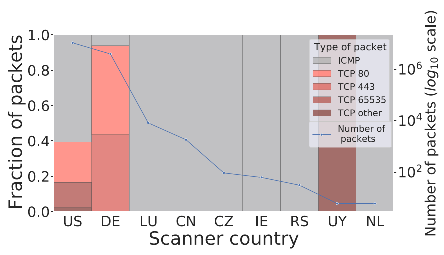
Table 1: Activity observed by our address space before and after experiments began from PCAPs and DNS logs.

3.2 Prevalence of scanning

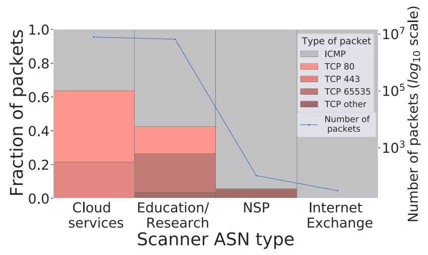
During the 48 week period of our study, our subnet received 14.6M packets and 699K DNS queries from 1068 traditional IP scanners and 96 NXDOMAIN scanners, respectively. We provide a summary of these scans in Table 1. Table 1 shows the number of IP and NXDOMAIN scans observed by our subnet before and after our experiments began, split by the treatment and control subnets. We see that, both control and treatment subnets appear to receive only small amounts of scanning activity per day before our experiments began in February 2021 from NXDOMAIN and IP scanners. Following the start of our controlled experiments, we see large increases observed in both groups of subnets — with some days receiving upwards of 100K IP scans and 1K NXDOMAIN scans. The plots are not normalized by the number of subnets. Figure 2 shows the amount of scanning activity destined for each of our 256 /64 subnets. First, we see a clear difference in the way IP scans and NXDOMAIN scans are used. Specifically, NXDOMAIN scans are narrowly focused on a smaller region of our address space while IP scans broadly target the entire /56 region. This attests to how the NXDOMAIN semantic can be used to discard entire subtrees and focus scanning attention on regions with active /64 subnets. Second, we see that not all subnets receive the same amount of attention from scanners — some receive either an increased number of scans or are scanned for longer durations of time. Third, we see that scanning activity is not consistent, there are several quiet weeks even after the experiments begin. In the next sections, we investigate the causes for these differences — i.e., what attracts scanners to specific subnets? (§4) and how do scanners find active subnets and what happens once they find one? (§5).

3.3 Scanner characteristics

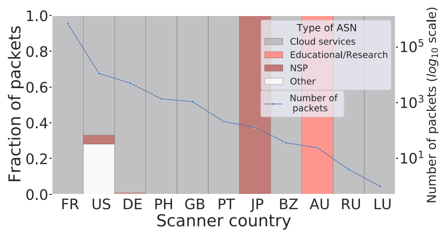
Scanner sources. We show the distribution of IP scanning traffic categorized by originating country (Figure 3a) and type



(a) IP scanning by country



(b) IP scanning by ASN type



(c) NXDOMAIN Scanning by country

Figure 3: Breakdown of scanning traffic by country and ASN type.

of AS (Figure 3b). Over 99% of all scanning traffic originated from just 2 countries, the US and Germany. Similarly, 99% of all scanning traffic originated from ASes belonging to cloud service providers or education/research institutes. We contacted the most active scanner in the education/research category of ASes to inquire the purpose of their scans. They responded that their goal was to "measure the state of IPv6 adoption". Other ASNs in this category belonged to research institutes including *China Next Generation Internet* which is tasked with with the "transition from IPv4 to IPv6" in China [4]. Of the scanning traffic that originated from cloud service providers, >99% of traffic originated from *Digital Ocean* and *ScaleUp Technologies*. Figure 3c breaks down NXDOMAIN scanning by country and AS type. We observe a similar pattern of two countries accounting for 99% of all NXDOMAIN scanning — France and the US. However, NXDOMAIN scanning originates predominantly only from ASes belonging to cloud service providers (> 99%), with 97% originating from the *OVH Cloud*. ASes belonging to education/research institutes comprise only a fraction of the total scanning traffic.

Scanner traffic characteristics and payload analysis. Figure 3a and Figure 3b also show the breakdown of types of scanning traffic by country and type of AS. ICMPv6 (46%) and TCP SYN (54%) scans dominate with almost all the scans comprising of these two types of traffic. Of the TCP traffic, three ports (HTTP, HTTPS, and port 65535) accounted for 97% of scanner probe targets. Of the 14 scanners that did not target these 3 services, 2 scanners conducted a port scan of the subnets running Tor relays. Services targeted by these port scans included Telnet, SMTP, FTP. The remaining scanners conducted a wider port scan targeting a total of 1,323 different ports. All 12 of these scanners originated from the same AS and sent very similar scanning traffic in terms of patterns in target addresses. These scanners targeted a total of 1,323 ports including those of popular services like Telnet, IMAP, PPTP, POP3, BGP. Of the 14.6 million scanning packets we captured, only 4K contained a payload. Packets that did not contain any payloads were either TCP SYN packets or ICMPv6 Echo Requests. Packets containing payloads were either HTTP or NTP packets. We analyzed the HTTP packets and found that one of the scanners had a distinctive `GET` request associated with an Nmap scan [8]. Further inspection showed that this scanner used Nmap to scan for potentially open services on port 80 like HNAP (Home Network Administration Protocol) with well-known exploits [21]. NTP packets did not seem to contain malicious payloads.

2013 [26]	2018 [36]	2020 [57]	2022 (this paper)
ICMPv6 Ping	HTTP (80)	HTTP (80)	ICMPv6 Ping
Port 7	ICMPv6 Ping	ICMPv6 Ping	HTTP (80)
HTTP (80)	-	HTTPS (443)	HTTPS (443)
SSH (22)	-	Redis (6379)	Port 65535
HTTPS (443)	-	6697 (IRC)	Telnet (23)

Table 2: Temporal differences in protocols targeted by IPv6 scanning (as observed in previous background radiation/scanning studies)

Presence of IP scanners on IP blocklists. To understand the intentions of the IP scanners, we examine whether they sent unsolicited traffic to other IP addresses on the Internet. We analyzed if the addresses of scanners targeting our address space were present on IP blocklists. We used AbuseIPDB [1] which maintains reputation of IPv6 addresses. Out of the 1,068 scanners observed in our measurements, 12 were reported for potential abuse on AbuseIPDB. 1,037 of the scanners originated from the `/32` IPv6 allocation of the AS *Alpha Strike Labs GmbH* which conducts global scans for research. The 12 scanners that were present on the blocklist contributed 51% of total IP scanning traffic. For 5 of these, we received their scans before they were reported on AbuseIPDB. Other scanners were reported on AbuseIPDB at most 3 months before they scanned our IPv6 subnet. All the reports submitted for these scanners mentioned unsolicited port scanning.

Temporal differences in targeted protocols. Table 2 shows the evolution of targets of IPv6 scanning over time. While the techniques used in previous work to detect scanning traffic in IPv6 might differ from ours, all studies capture traffic that is unsolicited/unclaimed. We observe that IP scanning traffic has shown similar characteristics over the years with ICMPv6 Ping, HTTP (80) and HTTPS(443) being the main targets.

4 IPv6 Scanner Address Discovery

In this section, we focus on answering *What address discovery strategies do IPv6 scanners use?* Put another way, what types of observable activities monitored by IPv6 scanners to discover IPv6 scanning targets.

4.1 Methodology

Overview. We allocate and start each of our services in a randomly selected group of four /64 subnets in a way that allows us to identify and isolate any causal effects that the services may have on IPv6 scanning (§4.1.1). Then, we use a *difference-in-differences* approach to test whether deploying a particular service has a causal effect on IPv6 scanning activity (§4.1.2). Finally, we measure the duration of any causal effects to uncover the typical lasting effect of deploying each of the services used in our study (§4.1.3).

4.1.1 Isolating the effects of services

Our large network allocation allows us to run a controlled experiment in which the 'liveness' of specific /64 subnets of our network can be leaked (by deploying services in them) for IPv6 scanners to find and the effects (i.e., the increase in scanning activity) of these leaks can be measured. We construct our experiment with the following key design decisions.

Preventing mixing of effects. Our experiment minimizes effects caused by two different types of services (e.g., Tor and NTP pool) from co-occurring. Not taking steps to explicitly prevent this may lead to incorrectly associating a causal relationship between a service and an observed effect. Our strategy to achieve this is to ensure that, at any point in time, only one type of service is deployed in our entire /56 allocation. This reduces the probability of several simultaneous effects that are caused by different services from being mistakenly considered as a single effect caused by one service.

Erasing the impact of internal residual effects. A residual effect in our setting is the increase in scanning activity that persists even after a service is discontinued. An internal residual effect is the effect that persists inside the /64 subnet that previously hosted a service. For example, scanners that learned of the liveness of a /64 subnet because of a Tor relay operating in that subnet may continue scanning it even after the relay is turned off, causing a lasting effect within the /64 subnet. Not minimizing the impact of residual effects due to previously deployed services can harm the accuracy of measured effect sizes for subsequently deployed services. It is reasonable to expect that the residual effects of a service are likely to be stronger inside the /64 subnet that hosted it than elsewhere in our /56 allocation. Therefore, we ensure that no single /64 is used by multiple services for the entire duration of this study. This prevents internal residual effects from impacting subsequently measured effects. We have two sets of services that seem as an exception to this rule, but are not. First, the 'OpenWPM' and 'wget' crawlers are two techniques for the same service and so were deployed in the same treatment subnets; and what we refer to as the 'NTP_{pool}' and '$\text{NTP}_{pool-20}$' NTP pool services are two different dates of the same NTP pool deployment — one when the NTP pool servers were actually deployed and one when the deployed servers were marked as stable and available for public use.

Reducing the impact of external residual effects. Residual effects from a prior service may also exist outside of its own /64 subnet. For example, a scanner that learned of the liveness of a /64 subnet because of a Tor relay operating in it may also scan neighboring /64 subnets for live hosts or services. Note that it is impossible to completely remove the impact of external residual effects — after all, we cannot know exactly which other /64 subnets a scanner might target after learning of a service in one. Therefore, we can only make a best-effort attempt to reduce the impact of such residual effects on our subsequent measurements of effect sizes. We do this by creating an upper-bound for external residual effects which in turn makes our subsequent measurements of effect sizes conservative. We achieve this by running each of our services for an extended period of time (*at least* two weeks). This allows any previously deployed services' external residual effects on the control region of our /56 to be reasonably integrated into the baseline (i.e., mean number of scans in /64 control subnets) upon which future service deployments may be compared with — thereby allowing for a conservative attempt at measuring future service effect sizes.

Reducing the impact of random effects. Since it is possible for scanners to chance upon one of our 'live' /64 subnets, we reduce the impact of random effects that may be unrelated to our services. We do this by deploying a single service (e.g., Tor relay) simultaneously on four randomly selected /64 subnets. Two of the selected /64 subnets are randomly chosen and have services deployed in the lower byte of their address space while the remaining two have service deployed at a random IP address contained in them. By measuring and reporting the average effect observed across all four of these subnets, we effectively reduce the impact of random effects. *These strategies improve: (1) our ability to correctly associate increases in scanning activity with specific services and (2) reduce the impact of latent confounding variables on the measured scanning activities.*

4.1.2 Identifying causal relationships and effect sizes

Our efforts in §4.1.1 allow us to mitigate the impact of many latent confounding variables that may affect the amount of scanning activity before and after a service is deployed, both inside or outside a service's /64 subnets. We use this controlled experimental setup to identify the increase in scanning activity as a result of deploying a service. We measure these effects using a difference-in-differences approach.

Obtaining control and treatment groups. We divide the 256 /64 subnets in our /56 address space into one *control group* ($\mathcal{C}$) consisting of 232 subnets in which no services were ever deployed and six *treatment groups* ($\mathcal{T}_{\text{crawl}}$, $\mathcal{T}_{\text{dns}}$, $\mathcal{T}_{\text{ntp-pool}}$, $\mathcal{T}_{\text{ntp-public}}$, $\mathcal{T}_{\text{tor}}$, and $\mathcal{T}_{\text{zone}}$) with each containing the four subnets allocated to the corresponding service. Recall, that no more than one of our treatment groups is active at any point in time and the intersection between any pair of treatment groups is null (§4.1.1).

Measuring narrow scanner effect sizes. We define a narrow scanner as one which is focused on identifying active hosts within a specific `/64` subnet. For such scanners, we expect to see an increase in scanning activity only within the treatment subnet and no effects in the control region. We measure their effect sizes using the techniques described below. Let t_s denote the time at which the service associated with treatment group $\mathcal{T}_s$ is deployed and $\mu_{\mathcal{S}}^{(t_i,t_j)}$ denote the mean number of scans observed per day between the days t_i and t_j per `/64` subnet belonging to group $\mathcal{S}$ (where $\mathcal{S}$ is the control group or one of our treatment groups). Now, the mean per-day difference in scanning activity occurring in $\mathcal{C}$ and $\mathcal{T}_s$ for the t days prior to the start of a service s (i.e., the last t pre-treatment days) is: $\Delta_{s,t}^{pre} = (\mu_{\mathcal{T}_s}^{(t_s-t,t_s)} - \mu_{\mathcal{C}}^{(t_s-t,t_s)})$. Similarly, the mean per-day difference in scanning activity occurring in $\mathcal{C}$ and $\mathcal{T}_s$ for the t days after the start of a service s (i.e., the first t post-treatment days) is: $\Delta_{s,t}^{post} = (\mu_{\mathcal{T}_s}^{(t_s,t_s+t)} - \mu_{\mathcal{C}}^{(t_s,t_s+t)})$ Then, we measure the *narrow scanner treatment effect size* over t days for the service s as: $\Delta_{s,t}^{\text{diff}} = \Delta_{s,t}^{post} - \Delta_{s,t}^{pre}$. Described in words, $\Delta_{s,t}^{\text{diff}}$ is the difference of the differences between the mean per-day scanning rates between $\mathcal{C}$ and $\mathcal{T}_s$ observed over t pre-treatment and t post-treatment days. An illustration of the measured treatment effect size is shown in the appendix (Figure 8). When Δ_s^{diff} is positive, it indicates increased scanning activity occurs inside the `/64` subnets which contain the service s. When Δ_s^{diff} is negative, it indicates that scanner activity is focused on the region outside the `/64` subnets — e.g., scanners may focus their attention on discovering other active subnets if they know a nearby subnet is live.

Measuring broad scanner effect sizes. We define a broad scanner as one which is focused on identifying active hosts in subnets that were not part of our treatment group. This is possible due to a different goal of identifying other live subnets rather than host IPs. Note that Δ_s^{diff} does not capture the increase in scanning activity caused by scanners which equally focus their attention on the control and treatment subnets. For example, if a scanner begins scanning our entire `/56` subnet space in a uniform fashion after discovering one of our treatment services, s, Δ_s^{diff} will be measured as 0. Therefore, we use a simple difference method, applied over all our control subnets, to measure the effects of these scanners. Specifically, we measure the *broad scanner treatment effect size* over t days for the service s as: $\Delta_{s,t}^{\mathcal{C}} = \mu_{\mathcal{C}}^{(t_s,t_s+t)} - \mu_{\mathcal{C}}^{(t_s,t_s-t)}$. Described in words, $\Delta_{s,t}^{\mathcal{C}}$ is simply the difference in the average scanning activity observed before and after a service s is deployed, computed over t days, and over all control subnets in our `/56` allocation. In the remainder of our reporting on the effect sizes Δ_s^{diff} and $\Delta_s^{\mathcal{C}}$ in §4.2, we set t as 14 days which is the duration of each service deployment.

Verifying statistical significance of measured effect sizes. To verify the statistical significance of the measured treatment effect sizes, we calculate bootstrap confidence intervals [27, 30] by resampling from the treatment and control subnets with replacement, using the percentile method to form a 95% confidence interval with the 2.5% and 97.5% bootstrap percentiles on 10K samples. We also calculate the p-values for a two-sided Welch's t-test [59] to test for a null difference between (1) $\Delta_{s,t}^{pre}$ and $\Delta_{s,t}^{post}$ in the case of narrow scanners and (2) $\mu_{\mathcal{C}}^{(t_s,t_s+t)}$ and $\mu_{\mathcal{C}}^{(t_s-t,t_s)}$ in the case of broad scanners.

Service	DNS logs		PCAPs	
	Δ_s^{diff}	$\Delta_s^{\mathcal{C}}$	Δ_s^{diff}	$\Delta_s^{\mathcal{C}}$
wget	**511.1**	-2.9	0.0	0.0
OpenWPM	**564.0**	-0.1	-0.4	0.4
DNS probes	**736.0**	1.9	**128.8**	**265.1**
NTP$_{pool}$	**348.3**	**6.0**	**313.6**	**734.4**
NTP$_{pool-20}$	**6.5**	**-9.7**	636.9	0.0
NTP$_{public}$	**72.2**	1.6	**116.3**	0.0
Tor relay	**87.1**	-0.4	0.0	0.0
DNS Zone	**1.2**	-1.3	-54.4	**588.1**

Table 3: Treatment effects observed in DNS logs and PCAPs. Services with a statistically significant Δ_s^{diff} or $\Delta_s^{\mathcal{C}}$ ($p < .05$) are in **bold**. *Cf.* §4.1.2 for details on Δ^{diff}, $\Delta^{\mathcal{C}}$, and significance testing.

4.1.3 Measuring internal residual effects

Since we do not reuse subnets in any of our treatment groups, we can continue monitoring them for any residual effects that remain after the service within it is no longer active. We measure these internal residual effects by measuring Δ_s^{post} over a 14-day sliding window for a period of 90 days after the service is halted. Lower values indicate a return to normal scanning activity, and indicate that the $\mathcal{T}_s$ subnets are received little to no additional scanning activity than the $\mathcal{C}$ subnets. This monitoring gives us an insight into scanner probing strategies — particularly, the freshness of their target lists.

4.2 Results

We first present the results of our causal analysis to identify treatment effect sizes caused by IPv6 NXDOMAIN scanners (§4.2.1) and all other IPv6 scanners (§4.2.2). Next, we report the internal residual effects caused by these scanners (§4.2.3).

4.2.1 NXDOMAIN scanning treatment effects

The effects measured by the deployment of each of our services, on scanners performing IPv6 NXDOMAIN scanning, are shown in the DNS logs columns of Table 3. Interestingly, we found that all scanners identified to be using the NXDOMAIN scanning approach may be categorized as narrow scanners — i.e., they primarily focused their scans inside our treatment subnets and had little to no scanning activity on our control subnets (characterized by high Δ_s^{diff} and low $\Delta_s^{\mathcal{C}}$). All services were found to have a statistically significant Δ^{diff} effect, although varying widely in their magnitudes.

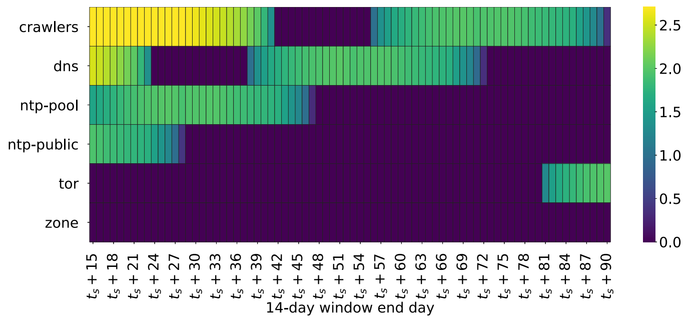
Effects of direct contact with potential scanners. The largest Δ^{diff} effect sizes were observed by our DNS probes service which sent DNS requests to a large number of IPv6 open resolvers. The effects of this service (*cf.* Figure 10a in appendix) show an immediate rise in IPv6 NXDOMAIN scanning activity which exclusively focusses on the subnets which sent the DNS requests (i.e., have low Δ^{C} values). Our crawling services, wget and OpenWPM, have similarly large effects with Δ^{diff} of 511 and 564 scans/subnet/day, respectively.

Effects of indirect contact with potential scanners. We also find that the two services, Tor and NTP, which cause our IP addresses to get added to archived public lists result in a delayed response from IPv6 scanners (*cf.* Figure 10b and Figure 10c in appendix). This behavior is expected due to two reasons: First, both services need participants to demonstrate a measure of reliability before they are publicly listed in the consensus as usable for Tor and NTP users. Second, unlike some of our other services (e.g., the DNS probes and crawlers), we are not directly interacting with potential scanners and instead are listing ourselves at a place where scanners may be monitoring at a pre-determined frequency. Next, we see that the initial deployment of the NTP$_{pool}$ service results in a significant increase in scanning activity with the /64 subnets, but achieving enough reliability to be promoted into the public NTP pool only results in a marginally increased scanning rate (an additional 6.5 scans/subnet/day greater than without the promotion). This is expected since our servers' promotion within the NTP pool would be completely inconsequential to scanners seeking to discover active IPv6 subnets. Finally, we see that our entries in zone files had close to no impact on the NXDOMAIN scanners with a Δ^{diff} of only 1.2.

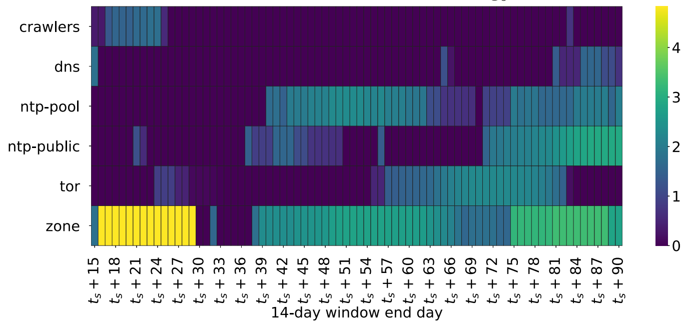
4.2.2 IP scanning treatment effects

The effects of our service deployment on the number of scans observed directly by our /56 allocation are shown in the PCAPs column of Table 3. Three of our deployed services (DNS, NTP$_{pool}$, and NTP$_{public}$) have a large and statistically significant Δ^{diff} effect while three have a large and statistically significant Δ^{C} effect (DNS, NTP$_{pool}$, and Zone). NXDOMAIN scanners focus not only on the treatment subnets, but also on the subnets in the control region, suggesting an intention to identify in-use subnets rather than host addresses.

Effects of direct contact with potential scanners. Unlike the NXDOMAIN scanners, we find that our crawlers did not attract any attention from IP scanners. Both, OpenWPM and wget had marginal Δ^{diff} and Δ^{C} values of $\in [-0.4, 0]$. However, we find that our DNS probes to open resolvers generate increased scanning activity (*cf.* Figure 10d in appendix). Interestingly, we note that the increase in scanning activity is seen in both, the control and the treatment subnets (Δ_i^{diff} s 128.8 and Δ_i^{C} s 265.1). Further investigation shows that scanners first identified and probed the treatment region for two days before returning with probes for addresses located in



(a) NXDOMAIN scanner residues ($\log_{10}$ scale)



(b) IP scanner residues ($\log_{10}$ scale)

Figure 4: Measured internal residue (Δ_s^{post}) values over a sliding 14-day window starting from $t_s + 15$ of each treatment until $t_s + 90$ days. Higher values are indicative of focused scanning within the treatment subnet. Heatmap values are scaled by $\log_{10}$.

the lower bytes of all our 256 /64 subnets (*cf.* Figure 9 in appendix). This behavior is explained in detail in §5.

Effects of indirect contact with potential scanners. We find a significant uptick in IP scanning activity in the treatment subnets after the deployment of NTP$_{pool}$ and NTP$_{public}$ experiments. However, effect of the NTP$_{pool}$ treatment was not found to be statistically significant since scanner attention was focused on only one of our four treatment subnets. Our DNS, zone, and NTP$_{pool}$ experiments also generated significant effects in our control region with Δ^{C} values of 265, 734, and 588, respectively. Same as before, we found that scanners began their probes focused on the treatment subnets or their neighbors before expanding to further away subnets.

4.2.3 Internal residual effects

Residual effects from NXDOMAIN scanners. Figure 4a shows the internal residual effects within each of our treatment groups are high. Specifically, we see that all subnets except for the ones used for Tor, targeted by the NXDOMAIN scanners during their treatment period continue to be scanned for a prolonged period of time — in fact, our wget and OpenWPM treatment subnets are targeted by high daily volume of NXDOMAIN queries for two extended periods of time separated only by a 15 day interval. This regularity and intensity is interesting since these experiments aim to mimic a home network being used for standard web browsing. We hypothesize that many scanners rely on web traffic logs as data sources (as our crawler experiments might suggest) to identify active hosts in such networks. We notice no NXDOMAIN scanning residue in the DNS, NTP$_{public}$, and NTP$_{pool}$ treatment subnets 26-70 days after the start of their treatments.

Residual effects from IP scanners. Figure 4b shows the residual effects observed from IP scanners. Similar to the NXDOMAIN scanners, we find prolonged IP scanning focused on our treatment subnets for nearly all treatments with the exception of our direct contact treatments — the scans generated because of the crawlers and DNS probes appear to end within 25 days and 1 day after our last use of their treatment subnets, respectively. We find that our listing in public sources of IPv6 addresses (NTP, Tor, and zone files) appears to have a lasting residual effect on the treatment subnets. This is expected since their addresses are left to be discovered by scanners that eventually monitor archives of these lists.

4.2.4 Takeaways

Our direct contact services (crawlers and DNS), which mimic user activity in a home network, attract statistically significant amounts of attention from NXDOMAIN scanners but not from IP probing scanners. Services which contribute to public network protocols (NTP, Tor, and zone) generate large amounts of scanning activity from one or both types of scanners. Unexpectedly, NXDOMAIN scanners never generate significant amounts of traffic in the control subnets *during the treatment period*. This suggests that they are initially used to identify live hosts within a `/64` subnet rather than to identify other active `/64` subnets (low $\Delta_s^{\mathcal{C}}$ values in Table 3). In contrast, IP scanners are used for both purposes (mixed Δ_s^{diff} and $\Delta_s^{\mathcal{C}}$ values in Table 3). Finally, our study on residual scanning effects shows that the prior use of a `/64` subnet for specific purposes does impact the duration of residual scanning activity. Notably, we find that our direct contact services, which mimic home network activity, generate large and persistent volume of NXDOMAIN scans but almost no IP scanning for a prolonged duration after the subnet becomes inactive. On the other hand, hosting web services which result in addition to the zone files generates consistent and significant volume of IP scanning activity even 90 days after the service is halted.

5 IPv6 Scanner Behaviors

We now focus on the question: *Once scanners have discovered an active IPv6 subnet, what behaviors do they exhibit?*

5.1 IP scanning

Overview. We seek to answer three questions: (1) what is the dominant scanning strategy for target address generation (§5.1.1) — do scanners target lower-byte or random addresses? (2) does the discovery of a treatment subnet with an active end-host having a specific type of address (lower-byte or random IID) impact the amount of scanning activity the subnet receives? (§5.1.2) and (3) how does scanner behavior change in scope over time — e.g., do they start as narrow-scanners and expand to broad-area scanning? (§5.1.3)

5.1.1 Target address generation strategies

IP probe categorization. To understand scanner address generation strategies, we grouped each probe into two categories; probes targeted at lower-byte IID addresses (P_l) and probes targeted at random IID addresses (P_r). We placed a probe in P_l if the IID section of the destination address (bits 64-128 in our case) have *at least* 40 leading zeros. If the probe did not meet this condition, we automatically assign it to P_r. Therefore, all probes belong in either P_l or P_r. Although this categorization appears simplistic, we settled on it only after noticing that it was indeed the case that all probes from a scanner that were placed in P_r were pseudo-random at the nybble-level (i.e., had maximum entropy for the nybbles they varied indicating that if a scanner altered higher nybbles, they tried all values for that nybble with nearly equal probability).

Which addresses are the most common scanning targets? In all, we observed 12.2M (84%) probes from 696 (65.1%) scanners addressed to 4,359 unique lower-byte addresses. In comparison, we observed only 2.3M (16%) probes from 747 (70%) scanners addressed to 2,291 unique random IID addresses. *The average lower-byte IID address was the target of 347,625$\times$ more scans than the average random IID address.*

Which scanning strategy do scanners most frequently use? 693 (65%) of all observed scanners used an exclusive probing strategy — i.e., either only probing lower-byte IIDs or only probing random IIDs. Of these, 321 were exclusively lower-byte scanners and the remaining 372 were exclusively random IID scanners. The lower-byte only scanners generated only 4% of all probes, the random IID scanners generated only 5% of all probes, and the remaining 91% of probes were generated by scanners using a mix of both.

5.1.2 Impact of addresses of known active end-hosts

Probe and destination subnet categorization. To understand if scanner behavior depends on the locations of the known end-hosts, we categorize each of our 24 treatment subnets into *lower-byte* (S_l) or *random* (S_r) based on whether the services deployed in them were on a lower-byte or random IID. We then categorized each probe sent to a treatment subnet into four buckets: $P_l \to S_l$, $P_l \to S_r$, $P_r \to S_l$, and $P_r \to S_r$. Figure 5 shows how scanners of different sizes probe these subnets. Here, a point indicates that scanners that sent a total number of probes corresponding to the y axis sent x% of their probes to a treatment subnet of the type indicated by the color. The density shows the number of scanners with similar x, y.

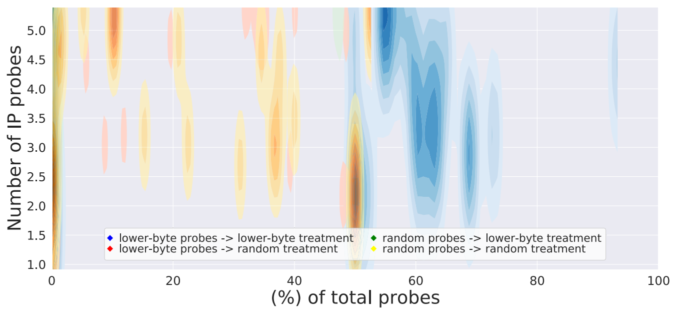


Figure 5: Types of probes sent to each type of treatment subnet by scanner sizes. (*Cf.* §5.1.2). Values are $\log_{10}$ scaled.

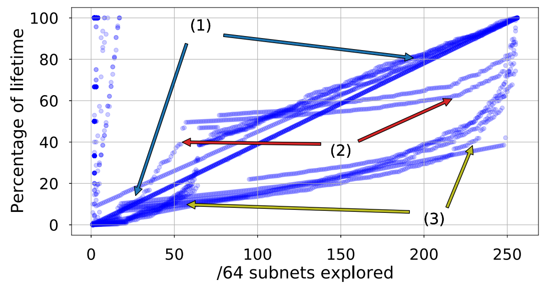


Figure 6: Subnets scanned over time by each IP scanner (§5.1.3).

Do scanners change their scanning behavior based on known host addresses? *In short, yes.* We focus on the most dense region of Figure 5 — i.e., the region with $x \in [35, 70]$. We make two observations that lead us to the above conclusion. First, scanners of all sizes send lower-byte probes to lower-byte treatment subnets (indicated by the blue density region) about 60-80% of the time. Second, when presented with a subnet in which they are aware of a random IID host, they send random probes (indicated by the yellow density region) nearly 50% of the time — about 3× more than the base rate. Further, no scanner ever sends random probes to a subnet in which the presence of a lower-byte host is known (the only green density region is at $x = 0$). For comparison, the base rate of random IID probes is 16%.

5.1.3 Change in scanner scope over time

Characterizing behavior by time. To understand how scanners change their scope over time, we analyze the number of `/64` subnets scanned by each scanner in their scanning lifespan (measured as the fraction of their total probes sent until that point in time). Figure 6 illustrates the breadth of scanning behavior over the lifetime of each scanner in our dataset. Each 'dot' in an (x, y) coordinate in the plot represents one point in the lifespan of one of the scanners and indicates that there was a scanner that, after having sent y% of their total probes ever sent, had probed x unique subnets.

Do scanners change their scope over time? *In short, sometimes.* From Figure 6, we observe four trends. First, on the top left we find that the narrow IP scanners do not change their behavior and focus on a handful of subnets over their entire lifetime. Then, we have one of three distinct behaviors that are observed in each of the broad IP scanners. These are annotated with (1), (2), and (3). The scanners exhibiting behavior (1) focus their scans equally on all our 256 subnets — sending the same number of probes in one before moving on to the next. These probes are mainly focused on the lower-bytes of each subnet. Scanners that form points highlighted by (2) initially focus their attention on a small number of subnets and then spread their probes broadly amongst the remaining. Nearly 40% of all their initial probes appear in just 50-60 unique subnets (which make up less than 25% of our address space) and then the remaining 60% of their probes sent later in time are distributed across the remaining 75% of subnets. Scanners making up the points marked by (3) show the opposite behavior in a more extreme form. They perform broad scans during the first part of their lifespan as they probe more than 200 unique subnets with just the first 40% of their probes before they focus their attention on the subnets that remain.

5.1.4 Takeaways

We see diverse scanning behavior in some dimensions and surprisingly uniform behavior in others. Our probe characterization shows that probes are most likely to focus their attention on lower-byte addresses in a subnet, but do not rule out probing random addresses. In fact, we see that probing behavior changes to incorporate more random scans in subnets where the presence of random addressing is already known to a scanner, while this never occurs when the presence of a lower-byte host is known. Finally, we see that there are just three distinct patterns observable in IP scanners that perform broad scanning — either uniformly broad scanning, or deep scanning for the first 40% of their probes followed by broad scanning with the remaining 60%, or vice-versa. We find that all scanners either focus on a very small number of subnets or perform scanning over all 256 subnets — never in the middle.

5.2 NXDOMAIN scanning

As discussed in §3.1, NXDOMAIN scanners can reduce the search space of finding active hosts significantly. This is achieved through scanning the *ip6.arpa* reverse DNS tree at different subnet length levels and discarding all subnet length trees under the current node if an `NXDOMAIN` response is received — after all, it is known that no domains exist within that entire subnet. This is the reason why our results in §4 shows that there is little to no NXDOMAIN scanning activity outside our treatment subnets. This property makes it necessary for us to use different metrics to characterize NXDOMAIN scanning. Therefore, we focus our analysis on understanding: when NXDOMAIN scanners know of the presence of a host with a particular address type (lower-byte or random) how do they scan the subnet to identify other hosts — do they assume lower byte nybbles at each level or

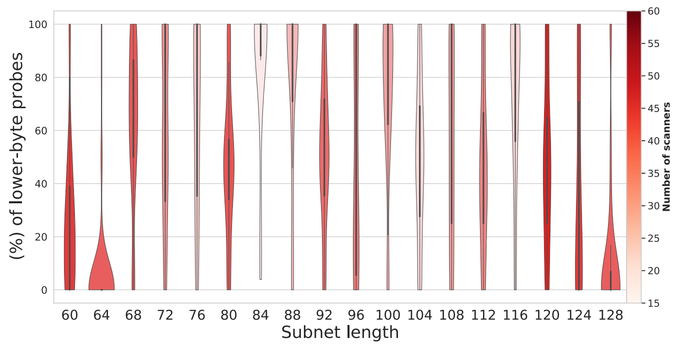


Figure 7: % of *lower-byte* NXDOMAIN scans at each subnet length.

do they branch diversely and does this change depending on the subnet level indicated in their probe?

Characterizing probe types in NXDOMAIN scanning. Probes generated by NXDOMAIN scanners can be of different lengths compared to probes generated by IP scanners which are always full 128-bit addresses. To understand probe generation techniques by NXDOMAIN scanners, we divide our probes into two categories; *lower-byte* and *random* . *lower-byte* probe of length *n* is a reverse PTR query for a subnet of `/n` that ends in 0. All non-zero queries are *random* probes.

NXDOMAIN scanning by probe type and subnet length. Figure 7 shows: (1) the number of scanners that sent probes for a subnet of the specified length and (2) whether the probes at this level were for the '0' subtree (i.e., a lower-byte probe) or '1-f' subtree (i.e., a random probe). Darker colors associated with the violin of a subnet length indicate more scanners sent probes at that subnet level. We find that most scanners focus their attention on `/60`, `/64`, `/80`, `/120`, and `/124`. This suggests that larger number of subnets expect hosts to be allocated within these subnet ranges. We now analyze the distributions (depicted by the violins) of lower-byte and random probes at each subnet level. The width of each violin is indicative of the number of scanners and that have *y*% of lower-byte probes (as a fraction of all their probes at that subnet level). Violins with wide areas at the bottom (e.g., `/64` and `/128`) suggest that scanners conduct mostly random scanning when sending queries at that level. On the other hand, smaller lower areas indicate more propensity for lower-byte scanning. We see that with the exception of the `/64` and `/128`, even NXDOMAIN scanners mostly rely on lower-byte probes (or have an expectation of host deployments at lower-byte addresses).

6 Security implications for network operators

In this section we discuss the implications of our findings for operators of IPv6 networks on the Internet.

Implications for address assignment in networks. We show that a lower-byte IID IPv6 address receives 347,625× the scans received by a random IID address (§5.1.1). Moreover, 99% of all IPv6 probes in our measurements targeted a lower-byte address. Therefore, we recommend network operators to use semantically opaque interface identifiers [39] for assigning IPv6 addresses to hosts in their networks. Such an address assignment will increase the scanning overhead by forcing scanners to probe 2^{64} addresses in the worst case and reduce the volume of probes received by an individual host.

Implications for allocating scanning defense budgets. We show that the volume of IP probe-based scanning activity, which is threatening due to direct interaction between the payloads and potentially vulnerable hosts, attracted by a subnet is highly dependent on the type of host activity in the network. This allows operators to consider allocating a security budget for their network that uses the expected host activities and addressing schemes in use to estimate scanning activity.

Implications for IPv6 subnet reuse. Our measurements show that operators can safely reuse subnets for different hosts in special cases without the risk of attracting scanners that targeted the previous host of the subnet (§4.2.4).

Implications on the utility of threat exchanges. Security practitioners have believed that increasing adoption of IPv6 on the Internet would reduce the the utility of Internet threat exchanges due to the infeasibility of blocklisting in the vast IPv6 address space [14, 20]. Blocking at the granularity of `/64` subnets increases the feasibility of blocklisting-based approaches with the potential of collateral damage due to hosts sharing fate on the same `/64` subnet [47]. We show that there is value in using blocklists for IPv6, at least in the current IPv6 landscape (§3.3). Majority of the scanning traffic observed by our subnet originated from IPv6 addresses already listed on popular threat exchanges like AbuseIPDB. Most of the scanners not present on the blocklist belonged to a Research/Education AS. Other scanners that sent significant scanning traffic while not being reported on the IP blocklist belonged to either a `/64`, `/48` or a `/32` shared by a scanner that was previously reported on the blocklist.

Implications for generating DNS PTR Records. The NXDOMAIN scanning technique (§2) can significantly reduce the number of probes required to find an active host in an IPv6 subnet. Thus, NXDOMAIN scanning can potentially bypass the security that semantically opaque identifiers would provide from active scanning. Dynamically generating PTR records when queried ("On the fly") as described in RFC 8501 [45] can render this scanning technique ineffective. We recommend network operators to use this technique since it enables generation of a valid PTR record for every address that is queried, preventing the scanner from exploiting the NXDOMAIN semantic to discard entire subtrees of addresses.

7 Related Work

In this section, we highlight the influence of previous research efforts on our study design and put our work in context.

IPv6 address discovery. Our decision to leak addresses through direct contact and indirect contact methods was informed by previous work by Gasser et al. [38]. Their work uncovers sections of the live IPv6 address space leveraged a variety of address sources. In their follow-up work [6, 37], they emphasized the importance of focusing address gathering on a wide range of sources and maximizing address stability rather than raw count when measuring efficacy of address gathering approaches. Our decision to include DNS logs in our data logging framework was informed by Fiebig et al. [5, 32, 33] who were the first to leverage the NXDOMAIN scanning approach described in §2 to identify unknown active IPv6 addresses. Their research has been incorporated into off-the-shelf IPv6 scanners. Expanding on the work of Fiebig et al., Borgolte et al. [24] use DNSSEC-signed reverse-zones to collect active IPv6 addresses and find it to work better than the NXDOMAIN approach for scanning.

Measurement of address allocation practices and IPv6 target generation. In our study, we deployed services with lower-byte and pseudo-random IIDs. This decision was inspired by the work of Gont et al. [40] who discussed patterns observed in IPv6 addresses collected from web servers, mail servers, clients and IPv6 routers observed in the wild. Their work showed that for each of these different categories, a majority the of IPv6 addresses follow a pre-defined address pattern — specifically lower-byte IIDs and pseudo-random IIDs. Focusing on measuring the stability of assigned addresses, Plonka et al. analyze active client addresses logged by a CDN [54]. They report that the IIDs of discovered addresses are unstable, but the `/64` subnets associated with them are stable. Along similar lines, Padmanabhan et al. conduct a temporal and spatial comparison of IPv4 and IPv6 addresses using 3000 RIPE Atlas probes [53] and find similar results regarding subnet stability.

IPv6 target generation. The findings from the above research on allocation practices have a direct impact on target generation algorithms used with IPv6 scanners. Foremski et al. leverage information-theoretic entropy at the nybble level to *mine segments* of the IPv6 address with similar entropy [35]. They apply this technique on addresses collected from a range of public data sources and find that client addresses have the most entropy in the last 64 bits whereas servers tend to use lower-bit addresses. Murdock et al. proposed *6Gen* [50] — an IPv6 target generation algorithm which uses a seed of input addresses to identify dense address space regions and determine patterns present in the seed addresses based on a probe budget. They find 56.7 million addresses in aliased regions and 1 million addresses in non-aliased regions.

Understanding scanner behavior. Ours is not the first work to shed light on the behavior of IPv6 scanners. In fact, these efforts date back to 2006 when Ford et al. [34] conducted the first measurement of IPv6 background radiation inside a darknet and found no evidence of scanner-related traffic. In 2010, Huston and Kuhne [46] repeated a similar experiment and discovered only a few thousand packets that were attributable to scanners (such as TCP SYN packets) and generally observed traffic from common IPv6 misconfiguration. A similar study was conducted by Czyz et al. [26] in 2013. They conducted a massive-scale measurement of IPv6 scanning by gathering and analyzing unclaimed traffic received by five `/12` IPv6 subnets. Despite their large vantage point into IPv6 scanning behavior, their study also only identified trace amounts of IPv6 scanning. More recently, in 2018, Fukuda and Heidemann [36] proposed using DNS backscatter to identify IPv6 scanning activity. Of note is the fact that this was the first study to find evidence of widespread IPv6 scanning activity. By analyzing the logs of an authoritative DNS server for reverse-DNS lookups, they identified (on average) 16 active IPv6 scanners per week over a six month period.

The work of Richter and Berger [56], while focused on IPv4 scanning behavior, informed our decision to conduct a controlled experiment with specific address leaks. Using a CDN as a vantage point, they find that different services attract scanners with different scanning strategies. Durumeric et al. [29] also observe similar patterns. They find that the services that scanners target change over time in IPv4; something we do not observe in IPv6. Like us, they also find that cloud service providers are the source of most of the scanning traffic.

While similar in goal, our study differs significantly from the highlighted IPv6 studies in two aspects: (1) IPv6 is much more ubiquitous today with nearly 35% global adoption than the time at which these studies were conducted (between 0.1% to 3%) [42] and (2) we take a unique approach by conducting a controlled experiment to specifically identify the behavior of scanners in response to specific types of services deployed in subnets (§4) and types of addresses allocated (§5).

8 Conclusions

We study IPv6 scanner address discovery strategies and the behavior of scanners when information about host deployment is known to them. Our controlled experiments reveal several key findings. First, we see different levels of focus and magnitude in the scanning that occurs in response to different types of address leakage — some resulting in the leaked subnets receiving, on average, 700 more probes/day/subnet while others result in all nearby subnets receiving large amounts in scanning. Next, we see that many scanners change their scanning behavior when a host in the subnet is already known to have a specific type of address. However, lower-byte scanning still remains the dominant scanning strategy even in subnets known to allocate random-IIDs to hosts. This results in lower-byte addresses receiving over 300K times more probes than random addresses, on average. Besides the key takeaway that *allocating lower-byte IIDs to end-hosts results in significantly increased risks because of scanning*, this study provides network operators with actionable insights regarding address allocation strategies to mitigate the impact of scanning and what to expect from scanners when a subnet is expected to run specific types of services.

Acknowledgments. We would like to thank the the ITS and CLAS Linux groups at the University of Iowa (JJ Urich, Jay Ford, Scott Allendorf, Matthew Brockman, Hugh Brown, Bradley Carson, Aran Cox, and Dan Holstad) for their help in setting up the experimentation testbed for this study. We would also like to thank our reviewers for their valuable feedback and suggestions. This research was funded by the National Science Foundation under award #1953983. The opinions, interpretations, conclusions, and recommendations pare those of the authors and are not endorsed by the funding bodies.

References

[1] Abuseipdb - ip address abuse reports - making the internet safer, one ip at a time. https://www.abuseipdb.com/. (Accessed on 06/06/2022).

[2] Github - lavalamp-/ipv666: Golang ipv6 address enumeration. https://github.com/lavalamp-/ipv666. (Accessed on 10/09/2021).

[3] Github - tumi8/zmap: Zmapv6: Internet scanner with ipv6 capabilities. https://github.com/tumi8/zmap. (Accessed on 10/09/2021).

[4] Iftf: Ipv6: China's next generation internet. https://www.iftf.org/future-now/article-detail/ipv6-chinas-next-generation-internet/. (Accessed on 06/01/2022).

[5] Ipv6 farm. https://ipv6.farm/. (Accessed on 10/09/2021).

[6] Ipv6 hitlist. https://ipv6hitlist.github.io/. (Accessed on 10/09/2021).

[7] Ipv6 scanning (-6) | nmap network scanning. https://nmap.org/book/port-scanning-ipv6.html. (Accessed on 10/09/2021).

[8] Ipv6 scanning (-6) | nmap network scanning. https://nmap.org/book/port-scanning-ipv6.html. (Accessed on 06/02/2021).

[9] Ipv6 toolkit – si6 networks. https://www.si6networks.com/research/tools/ipv6toolkit/. (Accessed on 06/02/2021).

[10] Ipv6 toolkit – si6 networks. https://www.si6networks.com/research/tools/ipv6toolkit/. (Accessed on 10/09/2021).

[11] pool.ntp.org: the internet cluster of ntp servers. https://www.ntppool.org/en/. (Accessed on 10/09/2021).

[12] pool.ntp.org: the internet cluster of ntp servers. https://www.pool.ntp.org/join/configuration.html. (Accessed on 07/13/2021).

[13] Sources – tor metrics. https://metrics.torproject.org/collector.html. (Accessed on 10/09/2021).

[14] Spamhaus ipv6 blocklists strategy statement. https://www.spamhaus.org/organization/statement/012/spamhaus-ipv6-blocklists-strategy-statement. (Accessed on 06/07/2022).

[15] Targeted ipv6 scans using pool.ntp.org . - sans internet storm center. https://isc.sans.edu/forums/diary/Targeted+IPv6+Scans+Using+poolntporg/20681/. (Accessed on 06/04/2022).

[16] The UCSD Network Telescope. https://www.caida.org/projects/network_telescope/. (Accessed on 10/10/2021).

[17] tumi8/zmap: Zmapv6: Internet scanner with ipv6 capabilities. https://github.com/tumi8/zmap. (Accessed on 06/07/2021).

[18] Webhome < servers < ntp. https://support.ntp.org/bin/view/Servers. (Accessed on 10/09/2021).

[19] What is ntp enumeration? - network time protocol - exploitbyte. https://exploitbyte.com/ntp-enumeration/. (Accessed on 10/09/2021).

[20] Will ipv6 end address blacklisting? - isc. https://www.isc.org/blogs/will-ipv6-end-address-blacklisting/. (Accessed on 06/07/2022).

[21] CVE-2021-34829. Available from MITRE, CVE-ID CVE-2021-34829., 6 2022.

[22] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. Apophanies or epiphanies? how crawlers impact our understanding of the web. In *Proceedings of The Web Conference 2020*, pages 271–280, 2020.

[23] T. Aura. Cryptographically generated addresses (cga). RFC 3972, March 2005.

[24] Kevin Borgolte, Shuang Hao, Tobias Fiebig, and Giovanni Vigna. Enumerating active ipv6 hosts for large-scale security scans via dnssec-signed reverse zones. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 770–784. IEEE, 2018.

[25] S. Bortzmeyer and S. Huque. Nxdomain: There really is nothing underneath. RFC 8020, RFC Editor, November 2016.

[26] Jakub Czyz, Kyle Lady, Sam G Miller, Michael Bailey, Michael Kallitsis, and Manish Karir. Understanding ipv6 internet background radiation. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 105–118, 2013.

[27] Thomas J DiCiccio and Bradley Efron. Bootstrap confidence intervals. *Statistical science*, 11(3):189–228, 1996.

[28] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, August 2004. USENIX Association.

[29] Zakir Durumeric, Michael Bailey, and J Alex Halderman. An internet-wide view of internet-wide scanning. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 65–78, 2014.

[30] Bradley Efron. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 82(397):171–185, 1987.

[31] Steven Englehardt, Christian Eubank, Peter Zimmerman, Dillon Reisman, and Arvind Narayanan. Openwpm : An automated platform for web privacy measurement. 2016.

[32] Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, and Giovanni Vigna. Something from nothing (there): collecting global ipv6 datasets from dns. In *International Conference on Passive and Active Network Measurement*, pages 30–43. Springer, 2017.

[33] Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, Giovanni Vigna, and Anja Feldmann. In rdns we trust: revisiting a common data-source's reliability. In *International Conference on Passive and Active Network Measurement*, pages 131–145. Springer, 2018.

[34] Matthew Ford, Jonathan Stevens, and John Ronan. Initial results from an ipv6 darknet13. In *International Conference on Internet Surveillance and Protection (ICISP'06)*, pages 13–13. IEEE, 2006.

[35] Pawel Foremski, David Plonka, and Arthur Berger. Entropy/ip: Uncovering structure in ipv6 addresses. In *Proceedings of the 2016 Internet Measurement Conference*, pages 167–181, 2016.

[36] Kensuke Fukuda and John Heidemann. Who knocks at the ipv6 door? detecting ipv6 scanning. In *Proceedings of the Internet Measurement Conference 2018*, pages 231–237, 2018.

[37] Oliver Gasser, Quirin Scheitle, Pawel Foremski, Qasim Lone, Maciej Korczyński, Stephen D Strowes, Luuk Hendriks, and Georg Carle. Clusters in the expanse: Understanding and unbiasing ipv6 hitlists. In *Proceedings of the Internet Measurement Conference 2018*, pages 364–378, 2018.

[38] Oliver Gasser, Quirin Scheitle, Sebastian Gebhard, and Georg Carle. Scanning the ipv6 internet: towards a comprehensive hitlist. *arXiv preprint arXiv:1607.05179*, 2016.

[39] F. Gont. A method for generating semantically opaque interface identifiers with ipv6 stateless address autoconfiguration (slaac). RFC 7217, RFC Editor, April 2014.

[40] F. Gont and T. Chown. Network reconnaissance in ipv6 networks. RFC 7707, RFC Editor, March 2016.

[41] F. Gont and W. Liu. A method for generating semantically opaque interface identifiers (iids) with the dynamic host configuration protocol for ipv6 (dhcpv6). RFC 7943, RFC Editor, September 2016.

[42] Google. Google ipv6 statistics. https://www.google.com/intl/en/ipv6/statistics.html. (Accessed on 10/10/2021).

[43] Luuk Hendriks, Ricardo de Oliveira Schmidt, Roland van Rijswijk-Deij, and Aiko Pras. On the potential of ipv6 open resolvers for ddos attacks. In Mohamed Ali Kâafar, Steve Uhlig, and Johanna Amann, editors, *Passive and Active Measurement - 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings*, volume 10176 of *Lecture Notes in Computer Science*, pages 17–29. Springer, 2017.

[44] R. Hinden and S. Deering. Ip version 6 addressing architecture. RFC 4291, RFC Editor, February 2006. http://www.rfc-editor.org/rfc/rfc4291.txt.

[45] L. Howard. Reverse dns in ipv6 for internet service providers. RFC 8501, RFC Editor, November 2018.

[46] Geoff Huston and Mirjam Kuhne. Background radiation in ipv6. *The ISP Column, APNIC*, 2010.

[47] Frank Li and David Freeman. Towards a user-level understanding of ipv6 behavior. In *Proceedings of the ACM Internet Measurement Conference*, pages 428–442, 2020.

[48] P. Mockapetris. Domain names - implementation and specification. RFC 1035, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1035.txt.

[49] T. Mrugalski, M. Siodelski, and B. Volz. Dynamic host configuration protocol for ipv6 (dhcpv6). RFC 8415, November 2018.

[50] Austin Murdock, Frank Li, Paul Bramsen, Zakir Durumeric, and Vern Paxson. Target generation for internet-wide ipv6 scanning. In *Proceedings of the 2017 Internet Measurement Conference*, pages 242–253, 2017.

[51] T. Narten and E. Nordmark. Neighbor discovery for ip version 6 (ipv6). RFC 4861, September 2007.

[52] Connection of IPv6 Domains via IPv4 Clouds. B. carpenter and k. moore. RFC 3056, February 2001.

[53] Ramakrishna Padmanabhan, John P Rula, Philipp Richter, Stephen D Strowes, and Alberto Dainotti. Dynamips: Analyzing address assignment practices in ipv4 and ipv6. In *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pages 55–70, 2020.

[54] David Plonka and Arthur Berger. Temporal and spatial classification of active ipv6 addresses. In *Proceedings of the 2015 Internet Measurement Conference*, pages 509–522, 2015.

[55] M Richardson, S Jiang, T Lemon, and T Winters. Internet engineering task force (ietf) t. mrugalski request for comments: 8415 m. siodelski obsoletes: 3315, 3633, 3736, 4242, 7083, isc. 2018.

[56] Philipp Richter and Arthur Berger. Scanning the scanners: Sensing the internet from a massively distributed network telescope. In *Proceedings of the Internet Measurement Conference*, pages 144–157, 2019.

[57] Stephen D Strowes, René Wilhelm, Florian Obser, Riccardo Stagni, Agustín Formoso, and Emile Aben. Debogonising 2a10::/12 analysis of one week's visibility of a new/12. In *Network Traffic Measurement and Analysis Conference (TMA)*, 2020.

[58] S. Thomson, T. Narten, and T. Jinmei. Ipv6 stateless address autoconfiguration. RFC 4862, RFC Editor, September 2007. `http://www.rfc-editor.org/rfc/rfc4862.txt`.

[59] Bernard L Welch. The generalization of 'student's'problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 1947.

A Additional Figures

Illustration of the measured effect sizes. Figure 8 illustrates the measured effect sizes from a service deployment inside a `/64` subnet.

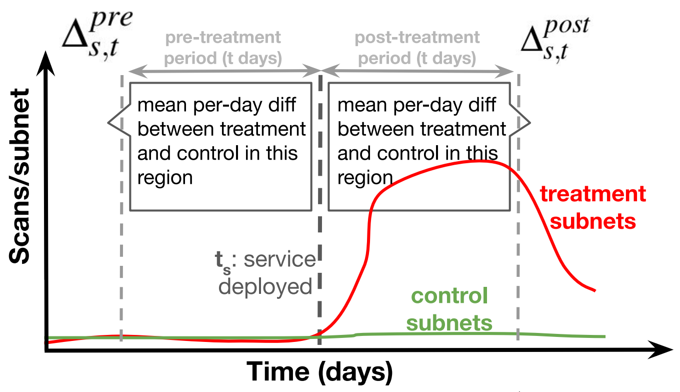

$\Delta^{pre}_{s,t}$
pre-treatment period (t days)
post-treatment period (t days)
$\Delta^{post}_{s,t}$
mean per-day diff between treatment and control in this region
mean per-day diff between treatment and control in this region
Scans/subnet
treatment subnets
t_s: service deployed
control subnets
Time (days)

Figure 8: An illustration of the measured $\Delta^{\text{pre}}_{s,t}$ and $\Delta^{\text{post}}_{s,t}$ for a service deployed at t_s (as explained in §4.1.2). The corresponding measured treatment effect size is $\Delta^{\text{diff}}_{s,t} = \Delta^{post}_{s,t} - \Delta^{pre}_{s,t}$.

Illustration of effects of direct contact with potential scanners. Figure 9 illustrates that scanners arriving during our (direct contact) DNS probe experiment first probed the treatment region for two days before returning to scan the addresses located in the lower bytes of all our `/64` subnets.

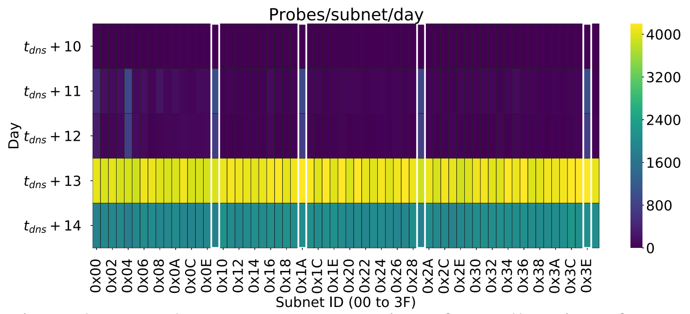
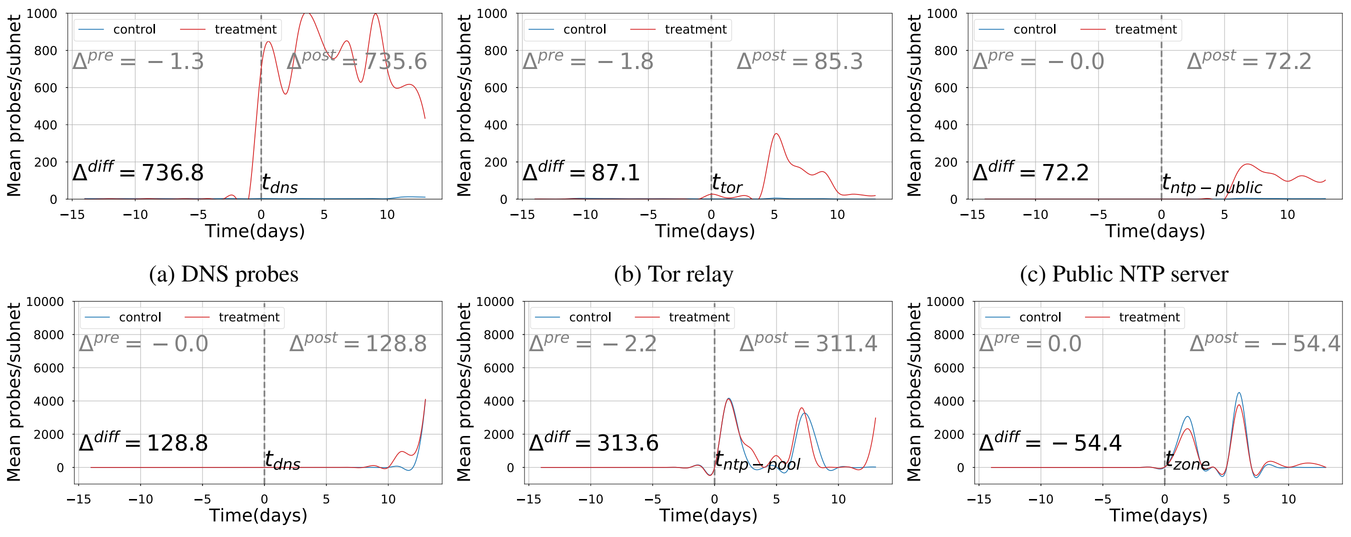


Figure 9: IP probes sent to a `/58` region of our allocation after deployment of our DNS probe service (corresponding to the day 10-15 region of Figure 10d). White highlighted cells indicate the subnets used for our DNS probe services (treatment group).

Selected illustrations of treatment effect sizes. Figure 10 illustrates the measured statistically significant treatment effect sizes observed from our experiments.



(a) DNS probes (b) Tor relay (c) Public NTP server

(d) DNS probes (e) NTP pool server (f) Zone file listing

Figure 10: Treatment effect sizes from deployed services observed in DNS scanning logs (10a, 10b, 10c) and PCAPs (10d, 10e, 10f). Only statistically significant effect are illustrated (Complete results in §4.2.1 and §4.2.2). We have applied linear interpolation to the visualizations for smoothing.