Is a Name Enough? A First Look into Detecting Clouds Using DNS Pointer Records

Sousan Tarahomi*, Raffaele Sommese*, Pieter-Tjerk de Boer*, Jeroen Linssen‡, Ralph Holz*†, Anna Sperotto*
*University of Twente, Enschede, The Netherlands, †University of Münster, Münster, Germany
‡Saxion University of Applied Sciences, Enschede, The Netherlands
{s.tarahomi, r.sommese, p.t.deboer, r.holz, a.sperotto }@utwente.nl, j.m.linssen@saxion.nl

***Abstract*—The flexibility and scalability of cloud services have led to their adoption across various industries. While it is easy to identify deployments of very large cloud providers (hypergiants) as they often publish the allocation of their network resources, this is much less commonly the case for smaller providers. Despite efforts by commercial IP intelligence providers to bridge this gap, it is not clear how complete and reliable their data is, which is compounded by the lack of transparency surrounding their identification methods. In this early study, we utilize reverse DNS, a public data source provided and used by network operators, to identify the IP cloud space. We develop a Markov chain-based classifier to identify patterns and structures in the reverse DNS naming schemes of cloud providers. Our results indicate that cloud infrastructure naming often differs significantly from that of residential IP space, although there are some overlaps that require further investigation. We believe that our model can be used by network operators to identify cloud deployments with varying levels of confidence.**

I. Introduction

The past two decades have witnessed a cloud computing revolution, making the term ubiquitous. Scalable and on-demand access to computing resources have driven widespread adoption across various industries. This has fostered a massive market dominated by giants such as Amazon, Google or IBM (to name a few) alongside numerous smaller providers that have embraced the flexibility and resource scalability of cloud computing as their core value [15].

This growing reliance on cloud services necessitates the ability to identify cloud deployments on the Internet and distinguish them from traditional infrastructure and end-user connections as a crucial factor in network security enforcement, traffic analysis and attribution, resource management, and content geo-blocking. For instance, streaming services need to ensure that the geographical distribution of cloud resources can be used to bypass geo-restrictions [10].

While some large cloud providers (e.g., Azure, AWS, Oracle) offer public IP lists of their infrastructure, aiding in identification, similar information for smaller providers is often limited or unavailable. Commercial services like IP2Location, IPInfo, and NetAcuity provide this information in commercial datasets, but their collection methods are generally business secrets.

This paper proposes an alternative approach for identifying cloud providers by leveraging the reverse DNS ecosystem. Managed by IP owners, reverse DNS offers valuable insights into an IP address's purpose. PTR records often reveal geographical location, infrastructure details, service types, and even provider identification. While past research has explored reverse DNS for network topology [8], [13] and privacy [16], we present a novel application: cloud deployment detection.

We investigate the structure of cloud resource PTR names and build a Markov chain-based classification system for cloud identification. Our results show that cloud deployments often exhibit distinct PTR structures compared to the residential IP space, enabling clear identification. However, we also identify instances of overlapping structures leading to misclassification. Our preliminary findings demonstrate that our model can be used with varying confidence, allowing operators to tailor cloud identification strategies for their network management needs.

II. Related Work

Several works in the literature focused on extracting information from Internet hostnames and, in particular, from the reverse DNS space. Arouna et al. found that around 40% of the allocated IPv4 address space has well-configured rDNS entries, with deployment patterns varying by region and driven by different factors in developed and developing countries [7]. Research has also focused on decoding rDNS naming patterns for broadband Internet end hosts, examining whether ISPs use intuitive keywords and if these keywords accurately reflect the underlying last mile technology [11]. Chabarek et al. developed PathAudit, a tool that extracts and decodes information from interface DNS names, revealing diverse encoding practices across networks [8]. Huffaker et al. developed DRoP, an automated system for geolocating routers by decoding geography-related strings in hostnames [9]. Luckie et al. further advanced this work by creating a system that automatically learns regular expressions to extract network names [14] and ASNs [13] from hostnames. Later, Luckie et al. further improved geolocation accuracy by training their system with geographic code dictionaries and constraining inferences with delay measurements. Their approach correctly geolocated 94.0% of router hostnames using geographic hints [12]. Reverse DNS records can also pose significant privacy risks by exposing client identifiers and network dynamics, potentially allowing device and individual tracking [16]. This abundance of information available in reverse DNS names prompted us

978-3-903176-66-9 ©2024 IFIP

to investigate the feasibility of identifying cloud deployments using reverse DNS names.

III. Methodology

The data input for our cloud detection model consists of reverse DNS PTR records. As mentioned before, previous research has shown that PTR records can encode, in various parts of the name, quite some information, such as, among others, geographical locations (e.g., a country, or a cardinal direction), infrastructural details about the host (e.g., if it is a virtual machine), or directly encoding an IP address. On a higher level, one can consider these details as a sequence of information tokens that together form PTR records. Given the sequential structure of the records, we chose to model them using Markov chains rather than machine learning (ML) methods. In previous experiments, we applied both clustering and classification algorithms but none of them performed better than Markov chain, since these traditional ML methods do not consider the sequential order of features while Markov chains are designed to model sequences.

A. Markov chains and likelihood ratio

A Markov chain is a set of states combined with a set of transitions (one-way arrows between states) that are labeled with the probability of moving between the states. In our context, the states are the parts of a hostname (called tokens, defined more precisely in Sec. III-C), and the transitions represent how the parts are put together sequentially to form a complete hostname. The transition probabilities are derived from a learning procedure based on a large set of example hostnames, see Sec. III-D.

For a given hostname and Markov chain model, multiplying the relevant transition probabilities gives the probability that this model would generate that hostname. This intuitively represents how well this hostname fits the model. However, this number *cannot* be interpreted as the probability that this hostname was generated by this model; that would require an a-priori probability distribution over various models, cf. Bayesian statistics.

What can be done however, is creating two Markov chains, for two different classes of hostnames (cloud and residential), and calculating the *ratio* of the likelihoods in both models. This ratio is a measure for whether the hostname fits one model better than the other. The optimal threshold may be far away from 1, however; e.g., if one model allows a much larger set of hostnames than the other, it will tend to give much lower likelihoods than the other, even for hostnames that do fit it.

B. Data Preparation and Tokenization

Before fitting the data including the PTR records of both cloud and residential explained in Sec. IV into our models, we prepared and tokenized them using automated scripts. We filter invalid and meaningless records (e.g., PTR records containing only IP addresses, PTR records without an SLDs).

PTR names are represented as subdomains separated by a dot (“.”). To find patterns inside these records, we removed the second level domain (SLDs) because they do not provide any information about naming convention patterns, but rather only indicate the organization owning the name. After removing the SLDs, we captured any type of IP representation within the PTRs records, including both IPv4 and IPv6 with different types of delimiters (“.”,“-”) and also partial IP addresses consisting of a subset of the rightmost octets of the corresponding IP address to the PTR record. To have a pattern of IP addresses, we replaced any IP representation with the string `ipaad`, a similar approach as in [11]. This pre-processing limits the variability introduced by IP addresses, while maintaining the structure of the PTR record. Next, to extract tokens from the PTR records, we split them based on both “.” and “-” delimiters. The reason to also consider “-” in addition to “.” is that, based on manual observation, we noticed that there is not only meaningful data in the entire subdomain but also within each subdomain, and similarly we can find meaningful data separated by “-”. Finally, we consider each part of the data as a separate token.

C. States Definition

The tokens offer us a way to group information within a set of PTR records in a meaningful manner. Analysis of the pointer records show that they can be grouped semantically in a small number of categories, which we use to define the state of the Markov Chain we are building. Based on manual analysis, we define the following states: *IP*, *geo*, *infra*, *digit*, *reg*, *none*. Additionally, we added two special states, *start* and *end*, to represent the beginning and end of the PTR string. The definitions of these states, which we also summarize in Table I, are as follows:

a) IP: Any representation of an IP address, replaced by `ipaad` string, is assigned to the IP state.

b) geo: The geo state refers to geolocation data. To detect if a token is a geocode, we check it against various types of geographical data: country, city, town, and region codes. We also consider geographical directions like “north”, and “southeast” as geocodes. For country codes, we used the ISO 3166-1 alpha-2 codes and for city and town codes, we used the GeoNames geographical database [1]. Finally, we manually defined a dictionary for geographical directions.

c) infra: We created a dictionary of meaningful words found in PTR records. This dictionary includes words like `dynamic`, `static`, `dedicated`, etc. Also we extracted the SLD name from the PTR (i.e., `example` for `example.com`), and assigned that token to the infra state, because we considered the SLD name as an indication the operator name. By checking the meaningful words in records, we captured the name of corporations and companies (e.g., Amentum), which we also added to our infra dictionary.

d) digit: Tokens consisting only of digits are assigned to the digit state.

e) reg: Based on our observations, many tokens start with characters followed by digits, such as vps1560. This pattern can provide information about the scope of the IP address, so we captured this regex pattern as the reg state.

TABLE I: State Names and Descriptions

State Name	Description
Geo	Geolocation code
IP	Any kind of IP representation for both IPv4 and IPv6 replaced by `ipaad`
Infra	A word from infrastructure dictionary
Digit	A string consisting of all digits
Reg	A regular expression starts with letters followed by digits (r'\^[a-zA-Z]+\[0-9]+')
None	None of the above states

f) none: Finally, if a token does not fit into any of the above states, it is assigned the state "none".

As an example, consider a PTR name with the value "ec2-100-22-74-198.us-west-2.compute.amazonaws.com". After removing `amazonaws.com` as the SLD, and replacing the IP address with the `ipaad` string, we obtain `ec2-ipaad.us-west-2.compute`. Fig. 1 shows the states for each token based on our algorithm.

$$\underbrace{\text{ec2}}_{\text{reg}} - \underbrace{\text{ipaad}}_{\text{IP}} . \underbrace{\text{us}}_{\text{geo}} - \underbrace{\text{west}}_{\text{geo}} - \underbrace{\text{2}}_{\text{digit}} . \underbrace{\text{compute}}_{\text{infra}}$$

Fig. 1: Markov chain states for a sample PTR

D. Transition Matrix

To build the transition matrix for the Markov chains, we first calculated state sequences. For each PTR record in our dataset, we calculated a sequence of states. This involved tokenizing the PTR record and assigning each token to a predefined state, based on our classification criteria. Then, we constructed 2-grams using the sequences of states. A 2-gram is a pair of consecutive states from the state sequence of a PTR record. For example in Fig. 1, we have (start, reg) as the first 2-gram in PTR. Finally, based on the 2-grams, we built the transition matrices shown in Table II and Table III. Each entry in the matrix represents the probability of transitioning from one state to another, calculated from the frequency of 2-gram occurrences across all PTR records. Namely, if ab indicates the 2-gram consisting of state a followed by state b, the transition probability p_{ij} of transitioning from state i to state j is

$$p_{ij} = \frac{|\{ab, a = i, b = j\}|}{|\{ab, \forall a, \forall b\}|} \tag{1}$$

E. Training, testing and evaluation

As indicated in Sec. III-A, a possible way of using Markov chains for classification is by training a model for each relevant class. In our case, we aim at distinguishing between cloud hostnames and non-cloud ones, which we refer to as residential hostnames. We therefore acquire two datasets, one for cloud PTR records and one for residential ones (see Sec. IV). We split both the cloud and residential datasets into 80% for training and 20% for testing and built two Markov chains from these two training sets. Then, for each sample in test sets, we calculated the probability that it belongs to the "cloud" Markov chain (p_{cloud}) and the probability that it belongs to the"residential" Markov chain ($p_{residential}$) using the corresponding chains built from training sets.

To calculate these probabilities, we first defined the states for each sample and then traversed from one state to another in the Markov chain, multiplying the transition probabilities obtained from the transition matrix. After acquiring p_{cloud} (Table II) and $p_{residential}$ (Table III), we calculated the ratio $p_{cloud}/p_{residential}$.

TABLE II: Transition probability for the cloud model

	start	ip	infra	geo	reg	digit	none	end
start	-	-	0.43	0.01	0.36	-	0.19	-
ip	-	-	0.39	0.18	0.02	-	0.08	0.34
infra	-	0.2	0.34	-	0.06	0.05	0.02	0.32
geo	-	-	0.07	0.3	0.06	0.46	0.1	0.02
reg	-	0.44	0.19	0.06	0.1	0.01	0.08	0.13
digit	-	-	0.66	0.03	0.01	0.01	0.02	0.27
none	-	0.33	0.09	0.16	0.03	0.02	0.11	0.26
end	-	-	-	-	-	-	-	-

TABLE III: Transition probability for the residential model

	start	ip	infra	geo	reg	digit	none	end
start	-	0.35	0.21	0.02	0.22	0.07	0.14	-
ip	-	-	0.26	0.07	0.16	0.03	0.32	0.17
infra	-	0.31	0.09	0.02	0.02	0.03	0.09	0.44
geo	-	0.03	0.06	0.16	0.01	0.07	0.11	0.58
reg	-	0.2	0.02	0.11	0.07	0.07	0.2	0.32
digit	-	0.24	0.08	0.01	0.05	0.37	0.18	0.08
none	-	0.2	0.05	0.07	0.01	0.04	0.14	0.49
end	-	-	-	-	-	-	-	-

IV. Datasets

The geolocation databases IP2location [2] and IPinfo [3] classify IP ranges in multiple categories, including the "cloud". Similarly to [18], we considered IP ranges with a "DCH" (Data Center/ Web Hosting/ Transit) usage_type in IP2Location and IP ranges with the hosting field marked as true in IPinfo as cloud. We created the ground truth for our Markov chain model by only taking the IP ranges where both geolocation providers agree. Additionally, we included samples from the published IP lists of three major providers: AWS [4], Azure [5], and Oracle [6] to have a broader range of cloud samples. For non-cloud samples, we again used the IP2location dataset, considering IP ranges with the usage_type "MOB", "ISP", and "ISP/MOB" as residential.

To extract the corresponding PTR records for these IP addresses, we used the OpenINTEL reverse DNS dataset [17].

The cloud dataset consists of 10,000 samples from each provider and 10,000 samples from the intersection of the geolocation datasets. To maintain a balance in our dataset, we selected 40,000 samples from the residential IPs. After preprocessing, to eliminate uninformative and invalid PTR records, we obtained 39,721 cloud samples and 39,161 residential samples (7,945 and 7,833 test samples).

V. Results

In this section, we present the outcomes of the analysis of our proposed models, in terms of classification results. We then explore the overlap between cloud and residential data sequences to better understand the capability of our models.

A. Classification results

We applied the trained models for cloud and residential PTR records to the test sets and calculated the ratio of cloud to residential probabilities. The ratio values obtained from these tests are shown in Fig. 2a, where the x-axis indexes the test samples. Fig. 2a can be roughly partitioned in three parts. On the top, we can see a prevalence of cloud samples, with a high probability ratio (> 100). Similarly, we see a band where residential samples are the most frequent ($< 10^{-2}$). In between, the two datasets overlap.

To gain a deeper insight into the distribution of the probability ratios, we plot the Kernel Density Estimate (KDE) for the probability densities of ratio values for the two test sets. Fig. 2b illustrates this. For better visualization, we presented the ratio values on a log10 scale. This scaling helps to highlight differences in the distributions more clearly.

Fig. 2b shows that cloud samples generally have a higher probability ratio than residential ones. The residential samples show a broader distribution of probabilities in our dataset, indicating that there is more variability in the type of sequences observed in this test set, and therefore more variability in the way the corresponding PTR records are built. We also see an overlap between cloud and residential samples. This represents the intrinsic limit of our classification, as for this overlap region our method cannot distinguish between cloud and residential PTR records (see Sec.V-B).

To use our model as a classifier, it is necessary to apply a threshold θ to the logarithmic probability ratio. If the probability ratio is larger than θ, the model identifies this as a "cloud" instance, otherwise a "residential" instance. However, the overlap region identified in Fig. 2b means that any chosen threshold will ultimately lead to false positives and false negatives. To assess the accuracy of our models, we plotted the Receiver Operating Characteristic (ROC) curve in Fig. 2c. The curve is obtained by calculating the true positive and false positive rates for $\theta \in [-10 : 0.5 : 4]$. The area under the ROC curve (AUC) is 0.93, showing our model has high accuracy.

B. Classification Overlap

As we saw in Fig. 2b, the logarithmic ratio distributions of cloud and residential sequences overlap. To gain a better understanding of this overlap, we analyzed the state sequences for samples with a log10(ratio) between -3 and 2. We identified 213 sequences from the cloud test set and 222 from the residential test set in this range. Among these, only 47 sequences were common to both cloud and residential datasets. Table IV presents seven of these sequences.

The observed shortest sequence length is three, namely for sequences in the form (start, state, end), where "state" is one of the states defined in Table I. Short sequences are often very generic, an observation that is confirmed by the fact that we identify five of these six sequences in the overlap (see Table IV). This means that, for these sequences, we are more likely to make a classification error. The sequences (start, reg/infra/none, end) appear in both cloud and residential samples with relatively high frequencies, and have a logarithmic ratio close to 0, where we observe the highest classification error. The exceptions are the sequence (start, ip, end), which is present in both datasets but with a clear prevalence among residential PTR records, and (start, digit, end), which is only present in residential samples. We also selected, as examples, three sequences that are unique to either the cloud or the residential category, based on their frequency, shown in Table V and Table VI, respectively, to highlight the uniqueness of those sequences.

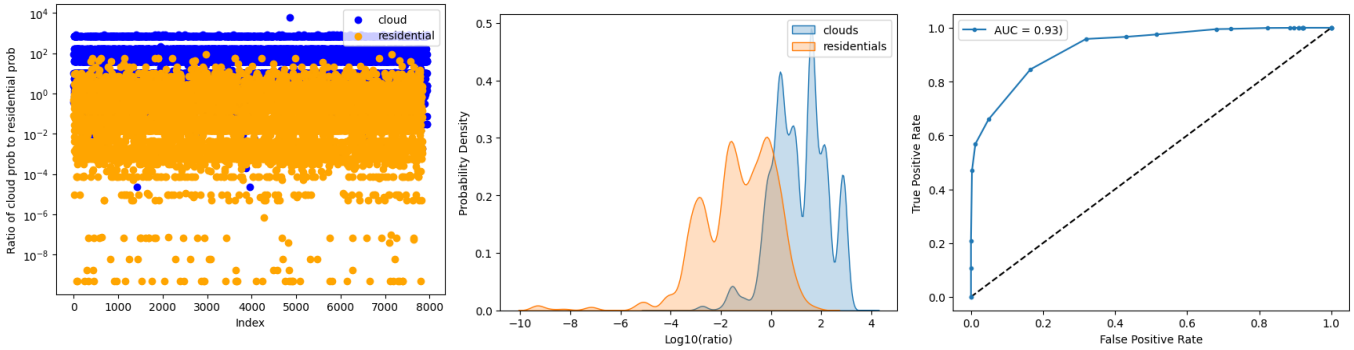
Interestingly, the top three highest values of unique sequences in cloud samples belong to two major cloud providers, Google and Amazon. Similarly, the second and third most frequent "residential" sequences belong to *cox.net* and *comcast.net*, respectively. We found several sequences that were specific to certain providers. For example, we found 36 unique sequences for AWS and four for Google, most of which exclusive to these providers. Also for residential samples, we found 11 and three unique sequences for *comcast.net* and *cox.net*, respectively. Similarly to cloud providers, they also have their exclusive sequences.

These findings suggest that while major providers can use similar naming patterns as others, they still employ some specific naming conventions that are unique to them. This characteristic can represent a good indicator to discriminate between clouds and non-clouds.

VI. Conclusion and Operational considerations

In this early study, we presented a Markov chain model based method to identify clouds based on DNS PTR names. We trained on ground-truth data provided by commercial services and major cloud providers. By defining distinct states derived from different types of information embedded in PTR records, we achieved a robust model with an AUC of 0.93, indicating high accuracy. Upon examining overlaps within our model, we found some common generic sequences, particularly single-word PTR names, shared across datasets, leading to misclassification. However, we also identified that major providers employ specific patterns exclusive to them. These unique sequences create discernible patterns for differentiation. We leave to network operators interested in detecting cloud deployments, the decision to establish the threshold for classifying cloud resources, recognizing that different FP and TP rates may reflect different operational needs. Based on the results of our analysis, we recognized that if cloud and residential operators marked their infrastructure more clearly by avoiding poorly descriptive names (i.e., resulting in short sequences), it would improve the overall transparency of address space usage.

Future research could improve our approach by expanding the PTR dictionary to include word variants and abbreviations,



(a) Ratio of cloud probability values to residential probability

(b) KDE plot of distribution ratio values

(c) Receiver Operating Characteristic (ROC)

Fig. 2: Classification results

TABLE IV: Common Sequences in cloud and residential IP ranges in the overlap.

Sequence	Samples in clouds	Samples in residential	Log10(ratio)	PTR Sample in cloud	PTR Sample in residential
(start, reg, end)	197	447	-0.167397	plesk03imp.imap.org.br.	softbank220031170212.bbtec.net.
(start, infra, end)	80	110	0.167701	mail.noicisiamo.net.	static.vnpt.vn.
(start, ip, end)	1	344	-1.554941	20-109-202-64.lot.net.	97e6b16a.skybroadband.com.
(start, none, end)	88	21	-0.137798	cccl.carlconnect.com.	subz.tmsasia.com.
(start, geo, end)	9	2	-1.658913	athens.billx.com.	kiboko.telkom.co.ke.

TABLE V: Unique sequences in cloud samples

Sequence	Samples	PTR Sample in dataset
(start, infra, infra, infra, ip, end)	1117	rate-limited-proxy-66-249-89-114.google.com.
(start, reg, ip, infra, digit, end)	551	ec2-3-229-215-54.compute-1.amazonaws.com.
(start, reg, ip, none, geo, digit, infra, end)	400	ec2-3-39-156-147.ap-northeast-2.compute.amazonaws.com.

TABLE VI: Unique sequences in residential IP range samples

Sequence	Samples	PTR Sample in dataset
(start, infra, ip, none, end)	305	net-5-94-85-17.cust.vodafonedsl.it.
(start, reg, ip, geo, geo, end)	208	ip98-177-18-173.ph.ph.cox.net.
(start, none, ip, reg, geo, end)	168	c-69-140-160-62.hsd1.md.comcast.net.

which were not explored in this study. This could further enhance the model's ability to accurately distinguish between cloud and non-cloud resources.

Acknowledgment

This research was funded by the Centrum voor Veiligheid en Digitalisering (CVD) and by the Netherlands Organization for Scientific Research (NWO) under grant number NWA.1215.18.003 (CATRIN). The research was made possible by OpenINTEL a joint project of the University of Twente, SURF, SIDN, and NLnet Labs, the operational support from the Twente University Centre for Cybersecurity Research (TUCCR). Finally, we thank IP2Location for the data access provided for this research.

References

[1] GeoNames. https://www.geonames.org/.
[2] IP2Location. https://www.ip2location.com/.
[3] IPinfo. https://ipinfo.io/.
[4] Published AWS IP list. https://ip-ranges.amazonaws.com/ip-ranges.json.
[5] Published Azure IP list. https://www.microsoft.com/en-us/download/details.aspx?id=56519.
[6] Published Oracle IP list. https://docs.oracle.com/en-us/iaas/tools/public_ip_ranges.json.
[7] A. Arouna, I. Livadariu, R. van Rijswijk-Deij, and M. Jonker. Advancing in reverse: A comprehensive characterization of in-addr. arpa deployment. In *AINTEC '23*, 2023.
[8] J. Chabarek and P. Barford. What's in a name? decoding router interface names. In *Proceedings of the 5th ACM Workshop on HotPlanet*, 2013.
[9] B. Huffaker, M. Fomenkov, and K. C. Claffy. Drop: Dns-based router positioning. *Comput. Commun. Rev.*, 2014.
[10] E. Khan, A. Sperotto, J. van der Ham, and R. van Rijswijk-Deij. Stranger VPNs: Investigating the Geo-Unblocking Capabilities of Commercial VPN Providers. *Lecture Notes in Computer Science*, 2023.
[11] Y. Lee and N. Spring. Identifying and analyzing broadband internet reverse dns names. CoNEXT '17, 2017.
[12] M. Luckie, B. Huffaker, A. Marder, Z. Bischof, M. Fletcher, and K. Claffy. Learning to extract geographic information from internet router hostnames. CoNEXT '21, 2021.
[13] M. Luckie, A. Marder, M. Fletcher, B. Huffaker, and K. Claffy. Learning to extract and use asns in hostnames. IMC '20, 2020.
[14] M. Luckie, A. Marder, B. Huffaker, and k. claffy. Learning regexes to extract network names from hostnames. AINTEC '21, 2021.
[15] N. Taleb and E. Mohamed. Cloud computing trends: A literature review. *Academic Journal of Interdisciplinary Studies*, 2020.
[16] O. van der Toorn, R. van Rijswijk-Deij, R. Sommese, A. Sperotto, and M. Jonker. Saving brian's privacy: the perils of privacy exposure through reverse dns. IMC '22, 2022.
[17] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras. A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements. *IEEE Journal on Selected Areas in Communications*, 2016.
[18] R. Yazdani, A. Hilton, J. van der Ham, R. van Rijswijk-Deij, C. Deccio, A. Sperotto, and M. Jonker. Mirrors in the sky: On the potential of clouds in dns reflection-based denial-of-service attacks. RAID '22, 2022.