Advertising DNS Protocol Use to Mitigate DDoS Attacks

Jacob Davis*
Sandia National Laboratories
Livermore, CA
jacdavi@sandia.gov

Casey Deccio
Brigham Young University
Provo, UT
casey@byu.edu

***Abstract*—The Domain Name System (DNS) has been frequently abused for distributed denial-of-service (DDoS) attacks and cache poisoning because it relies on the User Datagram Protocol (UDP). Since UDP is connection-less, it is trivial for an attacker to spoof the source of a DNS query or response. While other secure transport mechanisms provide identity management, such as the Transmission Control Protocol (TCP) and DNS Cookies, there is currently no method for a client to state that they only use a given protocol. This paper presents a new method to allow protocol enforcement: DNS Protocol Advertisement Records (DPAR). Advertisement records allow Internet Protocol (IP) address subnets to post a public record in the reverse DNS zone stating which DNS mechanisms are used by their clients. DNS servers may then look up this record and require a client to use the stated mechanism, in turn preventing an attacker from sending spoofed messages over UDP. In this paper, we define the specification for DNS Protocol Advertisement Records, considerations that were made, and comparisons to alternative approaches. We additionally estimate the effectiveness of advertisements in preventing DDoS attacks and the expected burden to DNS servers.**

***Index Terms*—Domain Name System, DNS, Identity Management, Protocol, DNS Cookies, DDoS Mitigation**

I. Introduction

The Domain Name System (DNS) has consistently been used for reflection-based distributed denial-of-service (DDoS) attacks which have the power to disrupt major internet infrastructure [1]. The DNS remains susceptible to these attacks because it relies on the User Datagram Protocol (UDP) for message transport. By design, UDP has no form of a connection and, thus, no method of verifying the sender of a packet. This makes it trivial for an attacker to spoof a query to a DNS server with the victim as the source, causing that server to respond and reflect traffic towards the victim.

Other transport mechanisms are supported in the DNS including DNS-over-TCP (Transmission Control Protocol) [2], DNS-over-TLS (Transport-Layer Security) [3], and DNS-over-HTTPS (Hypertext Transfer Protocol Secure) [4]. All three of these standardized mechanisms provide the identity management inherent in a TCP handshake and thus prevent spoofing attacks. DNS Cookies [5] are an alternative approach that adds unique identifiers to DNS messages. DNS Cookies allow UDP to be used while still preventing spoofing.

These protocols, and others, effectively enable a DNS client to ensure that the responses it receives are not spoofed. However, because they are client-centric, none prevent a reflection-based DDoS attack wherein the client's queries are spoofed. When an authoritative or recursive DNS server receives a query over UDP, it has no standardized method for determining whether that client uses UDP or an alternative protocol that includes identity management. As a result, a server will always respond to incoming queries—even those that may be spoofed.

In this work, we present DNS Protocol Advertisement Records (DPARs). Advertisement records enable subnets to publicly state the protocol used by their DNS clients in the corresponding address space by placing a DNS `TXT` record in the reverse DNS namespace. When receiving a query, servers that would otherwise act as reflectors can check for an advertisement and enforce that the given protocol is used. This effectively prevents any spoofed queries from being reflected by the enforcing servers towards a potential victim who has advertised its support of a more secure transport mechanism.

In §II we provide background information on the DNS, spoofing attacks, and alternative options for DDoS mitigation. Next, we define the specification for our protocol in §III. We then share our key design considerations and evaluate the protocol using data in §IV and §V. Finally, we conclude with limitations and future work in §VI.

II. Background

A. The Domain Name System

The Domain Name System (DNS) is an essential backbone of the Internet. Its primary responsibility is to translate domain names (e.g., `example.com`) to Internet Protocol (IP) addresses (e.g., `192.0.2.1`), but records containing other data are also used. The DNS architecture consists of two client-server pairs: stub resolvers communicate to recursive resolvers, and recursive resolvers communicate to authoritative servers.

While the DNS primarily translates domain names to other resources, the reverse DNS zone is used to provide records

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

* This work was completed while the author was a student at Brigham Young University.

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

for a given IP address [6]. This is often used for verification purposes. For IPv4 addresses, a specialized domain of `in-addr.arpa` is used for this resolution. The delegation of this domain follows that of IP addresses in reverse byte order. For example, any records for the IP address `192.0.2.1` can be found at `1.2.0.192.in-addr.arpa`. A similar implementation exists for IPv6 using the `ip6.arpa` domain.

Reverse records can also exist at standard subnet boundaries. For example, `10.4.0.0/16` could have a reverse record at `4.10.in-addr.arpa`.

B. Transport Protocols and Spoofing

UDP is the recommended transport mechanism given by the DNS specification [6]. However, TCP has also been a viable transport since the beginning [6], [2]. More recently, TLS [3], and HTTPS [4] have been established as additional transports. DNS-over-TLS and DNS-over-HTTPS have begun seeing adoption [7], but it is expected that effectively *all* DNS servers will continue to support UDP queries in order to be compatible with other clients and servers.

A major drawback of UDP is its lack of identity management. Unlike TCP (and TLS and HTTPS, which effectively extend TCP), UDP does not establish a connection prior to exchanging data. As a result, neither a client nor a server can verify the source of a received packet.

Lack of identity management with UDP enables spoofing, wherein an attacker falsifies the source IP address of the packet, effectively impersonating a third-party entity. In a cache poisoning attack, the attacker impersonates a server's response, and in turn, can redirect the client to the attacker's IP address. In a reflection-based attack, the attacker impersonates the victim's IP address and sends many DNS queries to recursive or authoritative DNS servers. This elicits activity from the DNS servers, resulting in the victim being flooded with unsolicited response traffic. Past attacks have reached traffic volumes of 300Gbps and are capable of affecting major services such as Amazon and Netflix [1]. These attacks are often referred to as *reflection* attacks because the malicious traffic is reflected off an unknowing DNS server. They are a type of distributed denial-of-service (DDoS) attacks because the result is an incapacitated victim.

C. DNS-Based Verification in SPF Records

Other protocols have used the DNS for out-of-band identity management. One example is the Sender Policy Framework (SPF) [8], which allows domains to publish a policy indicating which IP addresses can legitimately send email messages claiming to be from the domain. When a mail server receives an email, it looks up the associated domain's SPF policy to verify that the sender's IP address is among those allowed to send for the domain. If the policy is enforced, it will prevent an attacker from using the protected domain in the "from" address of an email, sending it from an illegitimate IP address, and having it perceived as legitimate.

A key aspect of SPF is that a DNS record is used as a source of information outside of the email exchange. This makes spoofing an email significantly more difficult as the attacker needs to either be on-path or needs to perform a successful cache poisoning to alter the policy that the mail server receives. We draw from this out-of-band communication method in the design of our protocol.

D. Related Work

A large body of research has been conducted to determine a method to prevent DDoS attacks. Mitigation methods typically fall into one of two categories: filters and challenges. Each method poses its own benefits and limitations.

Filter-based mechanisms have an advantage in that they do not require protocol changes; however, they rely on a system capable of handling attack traffic. For example, some solutions suggest filters at ISP edge routers [9]. Many large backbone providers also offer reverse-proxy services, essentially absorbing attack traffic with their high bandwidth [10]. These solutions are readily deployable and may be a good fit for entities that can afford them.

Challenge-based methods rely on clients verifying their identity by providing a unique token given to them by the server. A common example of a challenge is the sequence number of a TCP handshake [11]. An off-path attacker is unable to see the sequence number and cannot spoof packets as a result. DNS Cookies [5] are an alternative design that provides identity management at a strength similar to TCP but without the latency burden. They are included in DNS messages as a `COOKIE` option inside the Extended DNS (EDNS) `OPT` resource record [12]. Recent work [13] has found that around 30% of servers and 10% of recursive clients have adopted cookies since their standardization 5 years ago. A major issue with TCP, DNS Cookies, and other challenge methods is that they require server support and some mode of enforcement. As it currently stands, servers reflect spoofed UDP attack packets regardless of the client's preferred protocol [13].

III. DNS Protocol Advertisement Record Specification

We have noted that while protocols such as DNS Cookies, DNS-over-TLS, and DNS-over-HTTPS are seeing some adoption, no mechanism exists for enforcing their use. That is, servers must continue to answer DNS queries over unauthenticated UDP for all clients because a server has no mechanism for verifying a client's preferred protocol. As a result, clients and servers that use these secure protocols remain susceptible to spoofing attacks.

Here we present a new protocol: DNS Protocol Advertisement Record (DPAR). Advertisement records are designed to allow servers to enforce that a given client uses DNS cookies, or another protocol if that client has advertised support. With proper adoption, this would prevent reflection-based attacks: a server would know to refrain from answering a query lacking authenticated transport.

A. Protocol Overview

The advertisement protocol is designed to allow clients to protect themselves from reflection-based DDoS attacks. Clients publish a record stating their support of secure transport, allowing servers to drop incoming queries that are non-compliant with the advertisement. While the secure transport might be any one of the secure protocols aforementioned, we use DNS Cookies as the use case in this paper.

1) Threat Model: We assume a threat model where an active off-path attacker is capable of spoofing DNS packets with the victim as the source. The attacker's primary goal is to deny service to the victim by spoofing packets to a large number of DNS servers. We assume that each server behaves in a standard manner, implementing DPAR and other best practices (e.g., rate-limiting).

Advertisement records are about server-side authentication of DNS clients and do not provide additional protection from cache poisoning attacks. Other methods, such as DNS Cookies [5], `0x20` encoding [14], and source port randomization [15] have addressed client-side authentication of DNS servers. Further investigation of client-side authentication is outside the scope of this protocol.

Given that the primary motivation for advertisement records is in preventing denial-of-service (DoS) attacks, our threat model does not consider an on-path attacker. For an on-path attacker to be able to circumvent advertisement records for a large number of servers, they would need to be located near the victim's endpoint. However, an attacker in this position could simply drop traffic to and from the victim to effectively deny service. In other words, a much simpler method for on-path DoS already exists.

2) Client-Server Architecture: DNS queries operate on a client-server model. The two primary instances of this in the DNS are stub to recursive resolver and recursive resolver to authoritative server. In each case, the *client* is the one issuing the DNS query, and the *server* is the one receiving (and replying to) the query. Thus, a recursive resolver can play the role of either client or server, depending on the entity with which it is interacting. In the case of advertisement records, clients advertise their policy while servers (both authoritative and recursive) enforce the policies.

The advertisement protocol protects not only DNS clients but entire ranges of IP addresses. A record applies a policy to an entire subnet, preventing a spoofing attack against any IP in the range. As a result, any owner of IP space can benefit from the use of an advertisement, regardless of whether or not they maintain DNS clients. If the subnet does not have any DNS clients, a `none` policy can be applied—meaning that a server should not expect any DNS query activity from that IP space. In instances where a subnet contains only a single DNS client, the policy should match the client's behavior, but will additionally protect other IP addresses in the subnet.

Advertisement records must be enforced by servers, i.e., those receiving queries. Both recursive resolvers and authoritative servers should implement the protocol; enforcement by only one or the other leaves a set of servers available for reflection attacks. Enforcement requires servers to perform a lookup for advertisement records. While this is trivial for a recursive resolver (which acts as a client and server), it may require more significant changes for authoritative servers. One option may be for the authoritative server to work in tandem with a stub or recursive client that can retrieve records.

B. Protocol Specification

Any network advertises the use of a specific DNS secure transport protocol by including a record in their reverse DNS zone at the `/16` subnet level for IPv4 or the `/40` level for IPv6. For example, the record for `192.0.2.1` would be published at `0.192.in-addr.arpa` while the record for `2001:db8::1` would be published at `0.0.8.b.d.0.1.0.0.2.ip6.arpa`. If a single advertisement record does not apply to the entire subnet, exceptions can be made via advertisement *delegations* at individual `/24` or `/48` subnets. The discussion and analysis upon which the prefix lengths were selected for both IPv4 and IPv6 is found in §V-A.

A server (recursive or authoritative) supporting advertisements performs a reverse lookup for an incoming DNS query from a new client IP address. If an advertisement is found—whether at the base or the delegated subnet—the server enforces the policy by limiting its response to queries that conform with the policy. If an advertisement is not found at the base subnet (`/16` or `/40`), the server caches the lack of a policy and proceeds to behave like normal for the IP.

This effectively prevents a victim IP, who is advertising a secure protocol (e.g., DNS Cookies or TCP), from being attacked via reflection; all servers can check the policy and ignore any spoofed packets they see.

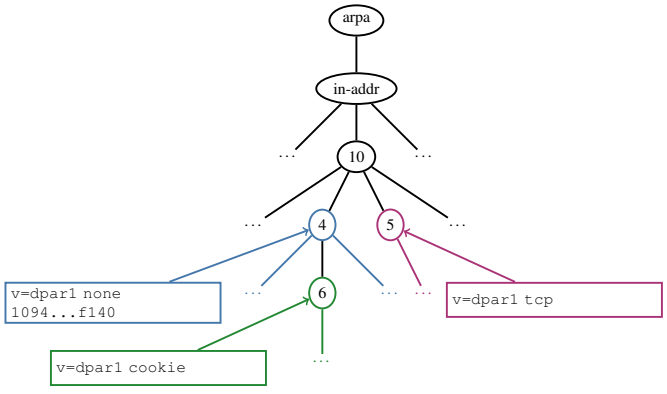

Fig. 1. An example of advertisement records in the reverse DNS for two IPv4 `/16` subnets (`10.4.0.0/16` and `10.5.0.0/16`) and a delegated `/24` (`10.4.6.0/24`). The hexadecimal string has been shortened for space. Like colors show which subnets are covered by the given policy.

1) Advertisement Record: An advertisement record must exist at the base `/16` or `/40` subnet level and can optionally delegate down to the `/24` or `/48` level. Figure 1 shows an example of delegation for an IPv4 network.

An advertisement DNS record is of type text (`TXT`) and has the format shown in Figure 2. The `version` field represents

⟨*dpar*⟩ ::= ‘v=dpar’ ⟨*version*⟩ ⟨*space*⟩ ⟨*transport*⟩ ⟨*delegation*⟩
⟨*version*⟩ ::= ‘1’
⟨*space*⟩ ::= ‘ ’
⟨*transport*⟩ ::= ‘udp’ | ‘cookie’ | ‘tcp’ | ‘none’
⟨*delegation*⟩ ::= ⟨*space*⟩ ⟨*hex-string-64*⟩ | ⟨*empty*⟩
⟨*hex-string-64*⟩ ::= ⟨*hex-char*⟩ × 64
⟨*hex-char*⟩ ::= ‘0’ | ‘1’ | ‘2’ | ...| ‘e’ | ‘f’

Fig. 2. Format of an advertisement record.

the version of the DNS protocol advertisement. Only version 1 is allowed at this time. The `<transport>` value represents the minimum level of security the subnet intends to use when sending queries; only a single value is specified (e.g., `udp` or `cookie`). The options shown are sorted by security level, from least to most secure. For example, a record with a `<transport>` value of `cookie` would allow cookies and DNS-over-TCP connections but not UDP. While we only specify options for DNS Cookies and TCP in this paper, future revisions might include DNS-over-TLS or DNS-over-HTTPS. We describe each option further:

- `udp`: DNS queries are expected to originate from this subnet, but there is no expectation of secure transport. This option exists as a formality and is the default level when no advertisement record exists. Clients may explicitly state that any mechanism is valid through the use of this record.
- `cookie`: Any DNS queries originating from this subnet will use DNS cookies or use TCP.
- `tcp`: Any DNS queries originating from this subnet will use TCP—or a higher security protocol if added.
- `none`: No DNS queries will originate from this DNS subnet; thus, any queries from this subnet can be considered illegitimate. This option allows clients to explicitly state that any DNS queries appearing to originate from the subnet should be dropped.

The `<delegation>` value is only valid in records published at the base level (`/16` for IPv4 and `/40` for IPv6). This parameter may optionally be added to a record in instances where the policies for some `/24`s or `/48`s differ from the base policy. It is composed of a 64-character hex string which is interpreted as a bit field. A bit value of 0 means to use the policy of the base record, while 1 means that the policy is delegated to the `/24` or `/48`. The right-most hexadecimal character represents the four least significant bits, i.e., bits 0 through 3. Delegation at other subnet boundaries should be achieved using multiple records or delegations (see §V-A).

Clients may choose the time-to-live (TTL) of the record as they would for any other DNS record. The recommended range of TTLs is between 3 and 48 hours. Clients may prefer a shorter TTL such that they are not locked out by servers in the case of a misconfiguration.

2) Example Record: We list three example advertisement records used in the IPv4 address space:

```
v=dpar1 none 1094cf9546e8a240511d946bc9d
    0400998800f4b11084881a80848612645f140
v=dpar1 cookie
v=dpar1 tcp
```

Figure 1 illustrates where the records might be placed in the DNS and which subnets they would affect if so placed. Specifically, the advertisement for `10.4.0.0/16` is published at `4.10.in-addr.arpa`, indicating that no DNS queries are expected from that `/16`. However, the delegation string indicates that policy has been further delegated to many of its constituent `/24` subnets, including `10.4.6.0/24`. The delegation to `10.4.6.0/24` is indicated by a value of 1 for bit 6, i.e., the only bit set in the right-most “40” in the delegation string. In the case of `10.4.6.0/24`, DNS queries are expected, but they must *at least* use DNS cookies. Finally, the policy for `10.5.0.0/16` is that all DNS queries from that `/16` will be over TCP, with no exceptions.

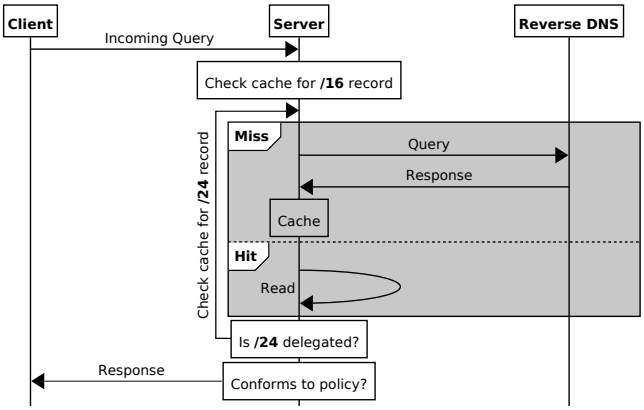

Fig. 3. High-level overview of the server enforcement process. If the server experiences a cache miss, it may respond normally to a predetermined amount of queries.

3) Server Behavior: Figure 3 provides an overview of a server’s behavior. Upon receiving a DNS query, a server checks for an existing advertisement record in its cache for the appropriate `/16` when the client is using IPv4 or the appropriate `/40` when the client is using IPv6. The server also checks its cache for a delegated record if (1) the base record is cached and (2) it has the delegation bit set for the client’s IP. If a cached record exists, the server must enforce the provided policy.

If a cached record does not exist, the server performs an advertisement lookup; however, this lookup does not need to be performed before responding to the incoming query. A server may define a maximum number of queries to respond to from a new client before an advertisement record is retrieved. This allows for the server to optimize its load without sending a significant amount of traffic to the potential victim.

If the server finds an advertisement record, it caches it per normal DNS operations. If the record requires delegation, the server performs an additional lookup for the `/24` or `/48` and caches that. A minimum TTL can be enforced by the server for any cached record to prevent excessively repeated queries.

If a record does not exist at the base level the server uses negative DNS caching [16] to prevent repeat lookups. Negative caching is also used when a record *does* exist at the base level, the policy is *delegated*, and the policy at the *delegated* subnet does not exist. A negative cache record, in either case, is the equivalent of a `udp` transport value. Negative cache entries should have a reasonable TTL. This allows for clients to add an advertisement record quickly and to have it distributed to all servers in a short time frame.

When a server is enforcing an advertisement policy and receives a non-compliant query, it drops the query. This prevents any attack packets from being reflected to the client. However, an exception must be made for enforcing a cookie policy. If incoming queries include a client cookie, but no server cookie, the server must respond with a valid server cookie (and optionally an answer) to a portion of the queries. This is consistent with the DNS Cookie specification. It ensures that a legitimate client is able to obtain a valid server cookie. However, these "bootstrap" responses must account for only a fraction of the total volume of incoming queries. Otherwise, the server could still be used effectively in a reflection attack.

IV. Design Considerations

A. Backward Compatibility and Incremental Deployment

Our protocol is designed to ensure backward compatibility with both clients and servers. In instances where a client publishes an advertisement record but a server does not support enforcement, operations proceed normally. The server will simply not query for the record and will respond to the client with no restrictions. In the opposite scenario, where a server supports enforcement but the client has no record, the server treats the client as having a `udp` policy (see §III-B). This policy is the least restrictive and enables the server to handle clients without an advertisement easily.

Built-in backward compatibility also facilitates incremental deployment. One approach to accomplish incremental deployment involves publishing a record with policy `udp` at the base `/16` or `/40`. As more specific prefixes desire to use advertisement records, they request to have the delegation bit set for the `/24` or `/48` subnets in the appropriate record, allowing them to manage their own policies. If all organizations within the base subnet later converge on a single, more secure policy, this policy can be placed at the `/16` or `/40` level and the delegations removed.

B. Authoritative Servers Performing Queries

This protocol requires authoritative servers to make outgoing queries in order to determine the advertisement policy of an incoming IP address. We acknowledge that this behavior is non-standard for authoritative servers since they do not have a client-side component like recursive resolvers. We suggest that an authoritative server utilize the stub resolver found on its system and preferably a local recursive resolver. Overall, a large and quickly accessible cache should be maintained. The implementation of that cache is outside the scope of this paper.

C. Shortcomings of Alternative Designs

While designing this protocol, we explored several alternative options. We provide an overview of several of these options below and why they ultimately were not chosen.

1) Records at Any Subnet: One option considered was to allow the advertisement record at any byte (IPv4) or nibble (IPv6) boundary (i.e., `/8`, `/16`, `/24`, and `/32` for IPv4 and `/4`, `/8`, ..., `/124`, and `/128` for IPv6). This provides far more flexibility and granularity for advertisers but increases the workload for servers. A server would need to check each subnet for a given IP to see if a record exists, requiring up to 4 queries for IPv4 and up to 32 for IPv6. Negative caching of an entire subnet would also be impossible because the lack of a record at an IPv4 `/16` does not prevent a record from existing at the `/24` or `/32`; the same goes for the IPv6 equivalents. Policy aggregation by subnet and negative caching are considered essential in our design, in part because a vast majority of DNS clients do not support secure protocols, and requiring a policy lookup for each new IP address would be burdensome. This claim is further addressed in §V-D.

2) Distributed Sharing between Servers: In another protocol alternative, servers would collect data about what protocol a client uses and then combine that data with that of other servers to form a stronger assumption of support. Because policies are inferred under this protocol, DNS clients would not need to advertise protocol use; A major shortcoming of this design is that an attacker could spoof UDP packets to trick servers into being unsure of the client's preference, an effective "downgrade" attack. This is true of any design which tries to infer client behavior. With the chosen design, a client explicitly publishes its policy, so there is no dispute. An explicit record also prevents issues related to Network Address Translation (NAT) wherein the clients behind a single, public IP address act independently and support a range of protocols.

3) Advertising Support within Queries: A final alternative would be to have clients advertise protocol support in their queries, e.g., as part of the EDNS record in a DNS message whose sender identity has been authenticated, i.e., with DNS Cookies or TCP. Servers would learn the capabilities of those clients via authenticated DNS messages from the clients themselves. There are several shortcomings of this alternative. First, the victim of a DNS reflection-based DDoS attack is not necessarily a DNS client. Therefore, servers may have never been contacted by the target IP address and would be unaware of its capabilities. Even in the case where a victim's IP address corresponds to a DNS client, many servers will be utilized in an attack, and it is unlikely that a given reflecting server will have been queried by the DNS client. It is imperative that a server be able to find a client's advertisement policy without requiring previous contact from the client.

V. Evaluation

In this section, we quantitatively analyze advertisement records by looking at data for the IP address space, queries to the authoritative root servers, and traffic for our organization's recursive and authoritative servers.

A. Administrative Feasibility of Records

One of the most substantial constraints we impose on this protocol was the prefix length associated with the base and delegated policies, i.e., `/16` and `/24` for IPv4 and `/40` and `/48` for IPv6. The decisions to use a `/16` and `/40` for the location of the base policy for IPv4 and IPv6, respectively, were made to balance between the total number of records a server may need to retrieve and the distribution of entities within a given prefix. In general, a shorter prefix covers more address space, and therefore, fewer queries are required to retrieve policies from within that address space. However, it is more likely that the address space covered by a shorter prefix is announced by more entities, making policy coordination more difficult.

We use Internet routing data to consider this balance and support our design decision. Using the IP prefix-to-autonomous system (AS) mapping from iptoasn.com [17], retrieved April 21, 2021, we analyze the distribution of prefix announcements, considering AS and prefix length. This distribution is shown in Figure 4. The average number of ASes announcing more specific IPv4 prefixes within a `/16` subnet is 8.4, with a median of 2 ASes. In other words, 8.4 different ASes, on average, announce IP prefixes within an IPv4 `/16`. We find this number small enough to be manageable from an administrative perspective. For example, if one AS desires to support advertisements in its prefix, a `udp` policy could be added at the `/16` with delegation to the AS's `/24`(s); alternatively, other AS prefixes could also adopt the policy, removing the need for delegation. For comparison, an average of 700 different ASes announce prefixes within a `/8` subnet—a number that appears much too large to effectively coordinate across organizations.

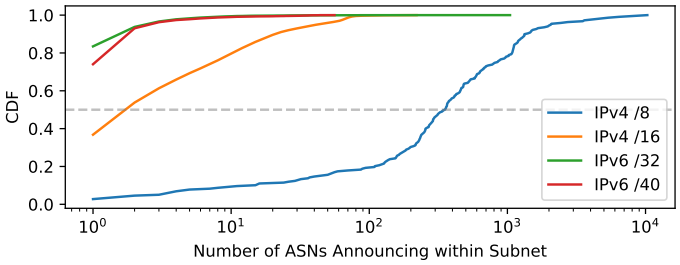

Fig. 4. Number of distinct ASes announcing more specific prefixes within IPv4 prefixes of length 8 and 16 and IPv6 prefixes of 32 and 40.

For IPv6, 74% of `/40`s and 83% of `/32`s have only one AS announcing their address space. This is characteristic of IPv6 addressing because the address space is so vast. Both `/32` and `/40` are therefore attractive options for publishing the base record (i.e., the equivalent of IPv4 `/16`). Notably, for nearly 100% of `/48` prefixes, only a single AS announces its address space (not shown in Figure 4); rarely is inter-organization coordination of policies a concern at the `/48` level.

While we consider the relatively low number of ASes announcing prefixes within an IPv4 `/16` to be manageable, other evidence supports our selection of a `/16` and `/24` being the optimal locations of the base policy and delegated policy, respectively. Figure 5 shows the distribution of sizes for announced IPv4 and IPv6 prefixes. The vast majority of announced IPv4 prefixes are between `/16` and `/24`, and nearly half of announced IPv6 prefixes are between `/40` and `/48`. Of the 1,995 IPv4 prefixes that are larger (i.e., contain more addresses) than a `/16`, 82% are either a `/14` or `/15`. As a result, most large prefixes would only need a maximum of 4 records to cover their entire range as the primary entry point for policy. Similarly, there are 309,000 announced IPv4 prefixes which are between `/17` and `/24`, 82% of which are either a `/22`, `/23`, or `/24`. Again, a vast majority could be covered by only a few delegated records.

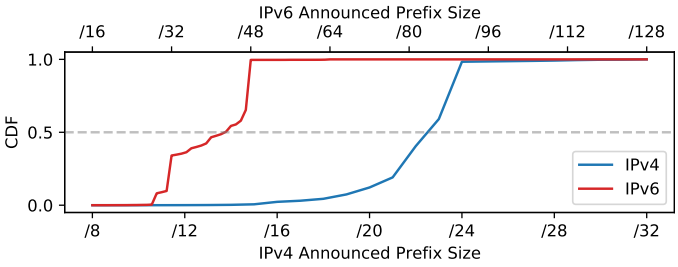

Fig. 5. Distribution of announced IPv4 and IPv6 prefix lengths.

For IPv6, only 10% of announced prefixes are larger than (i.e., have more addresses than) a `/32`. In other words, 90% of organizations announcing a prefix of size `/32` or smaller simply need to publish up to 256 records corresponding to the `/40` subnets comprising their subnet. IPv6 also shows similar characteristics to IPv4 for subnets between the base and delegated advertisement records (i.e., `/41`–`/48`). There are 25,987 prefixes in this range, and 83% are of `/46` or higher. So again, most IPv6 prefixes sized between records can be handled by only 4 records.

B. Estimating Advertisement Record Landscape

We continue our evaluation by estimating what the distribution of advertisement records may look like. Specifically, we are interested in seeing how many `/16` prefixes (or IPv6 `/40` prefixes) can be covered by a single record, and how many have to delegate.

For this analysis, we look at the day-in-the-life (DITL) data of the DNS root servers from 2020 [18]. This data captures all incoming queries to the root servers over a 48-hour period. These queries typically come from recursive clients. We note that there is no data available for b-root or g-root in the 2020 collection. Also, the data for i-root and l-root is anonymized and can therefore not be used in our analysis. We analyze the remaining 9 servers. We also note that some clients may run a local root server, and therefore, may never query the root servers [19]. Despite these limitations, the DITL data provides insight into some of the most commonly used authoritative servers. For each IP address that queried the root servers, we record the number of UDP queries sent, with and without cookies, as well as the number of queries made over TCP.

1) Number of Possible `none` *Records:* Our first interest is in seeing how many subnets do not ever produce queries to the root servers during the 48-hour collection period. Treating the DITL capture as a representative sample, such subnets could easily be covered by a single `none` advertisement record.

We initially find that at least one DNS query originates from 38,002 (58%) of all IPv4 `/16`s — with a median of 60 querying IP addresses per `/16` that produced a query. Thus, we immediately infer that 42% of `/16`s can be covered by a single `none` policy since they have no DNS clients. Once we consider delegation, we find that 83% of `/24`s do not send a query and can therefore use a `none` policy.

For IPv6, DNS queries are observed from 56,891 `/40` subnets. While this is similar to the number of IPv4 `/16`s, it represents a near-zero amount of the total IPv6 address space. Covering the remaining IPv6 space by publishing `none` policies at every `/40` is infeasible. Fortunately, that is not necessary. Advertisement records are published by a network entity to protect that network entity from becoming the victim of a reflection attack. As was shown in §V-A, 90% of organizations can cover their entire network in 256 advertisement records or less.

2) Number of Delegated Records: We next consider how many instances of delegation will be necessary, i.e., from `/16` to `/24` (IPv4) and from `/40` to `/48` (IPv6). To do this, we analyze the DITL data to make assumptions about the current capabilities of DNS clients and how policies might be used if they are based on that behavior.

First, we record the highest level of security transport exhibited by each querying IP address, specifically UDP with cookies (`cookie`) or UDP (`udp`). In the case that we observed only TCP queries from a given client, we could not infer whether or not the client supported cookies (because DNS over TCP does not use cookies). Relatedly, we exclude TCP-only (i.e., policy `tcp`) as a secure transport policy; the mere presence of a TCP query does not indicate that a given client is ready to deploy a TCP-only query strategy—even if we only observe TCP queries in the DITL data—nor does the lack of a TCP query imply that the DNS client is incapable of querying over TCP. The DITL data is simply insufficient for inferring more than that. Finally, there are multiple reasons that we might observe multiple transport capabilities from a given IP address, one prominent example of which is NAT. For the purposes of this paper, our heuristic only considers the highest level of security for a given IP address.

Having identified the most secure DNS transport exhibited by a given IP address, we label each delegated (i.e., `/24` or `/48`) with a secure transport policy, basing it on the capabilities of its constituent IP addresses. A delegated (`/24` or `/48`) subnet is labeled with a `none` policy if no DNS queries were observed from that subnet. In the case that at least one query was observed, there are two potential strategies for estimating the policy for a given `/24`: either use the *dominant* transport or the *least secure* transport. The dominant transport policy follows the line of thinking that the minority of resolvers in that subnet could be "upgraded" if the majority already is, such that the delegated subnet could publish the more secure policy. On the other hand, the least secure policy is what must be published if they cannot be upgraded. For our analysis, we estimate policy for a given `/24` using the dominant transport.

Next, we extend our policy estimation to the base subnet level (i.e., `/16` or `/40`). We estimate a policy for each base subnet based on the dominant policy applied to the collective delegated subnets within its address space. In the case of the base subnet, using the dominant strategy is most efficient (as compared with the least secure strategy). Every subnet with a policy different from that of the base subnet will be delegated to, such that it can publish its own policy, overriding that of the base. Thus, the dominant strategy approach minimizes policy delegations.

We first present the analysis resulting from applying policies at the IPv4 `/16` level according to the strategies previously described. As mentioned previously, no query was observed from 42% of `/16`s, indicating that they can adopt a `none` policy. An additional 47% have a dominant policy of `none` (since no queries were observed from a majority of their `/24`s); for the remaining 7,508 (11%), the dominant policy is `udp`.

There are a total of 2,160,807 `/24`s for which the policy at their parent `/16` is different than that at the `/24` itself; that is an average of 57 delegations per `/16` for which at least one query was observed. If we include the 2^{16} records published at the base `/16` level, approximately 2.2 million advertisement records would be necessary to cover the entire IPv4 address space in the state we analyzed. It is important to note that configurations may change if advertisement records are adopted. This could increase or decrease the total number of records required.

Looking at IPv6, we observe only 22 `/40`s for which the dominant policy was `udp`. The remaining subnets had a dominant policy of `none`. While the frequency of `none` exceeds that of `udp` in both cases, it is much more dominant with IPv6, with effectively 100% of dominant policies being `none`. In the case of IPv6, only 193,680 delegation records would be required, an average of only 3.4 delegations per IPv6 `/40` for which DNS activity was observed.

3) Uniformity within Delegated Subnets: The advertisement protocol record is designed to have a maximum precision of a `/24` (IPv4) or a `/48` (IPv6). Therefore, it is necessary for all IPs within a delegated subnet to follow the same policy. We examine the uniformity within `/24`s and `/48`s to determine whether a record can apply to an entire subnet.

We first consider the simplest cases, where there is either no DNS activity within an IPv4 `/24` or the only activity comes from a single active IP address. As we previously found, 83% of IPv4 `/24` subnets produced no DNS activity and can thus be handled with a `none` record. For IPv6, all but 195,036 `/48` subnets can be handled by a `none` policy. For 35% of the remaining `/24` subnets and 59% of the remaining `/48` subnets, DNS queries were observed from only a single IP address. For these delegated IPv4 and IPv6 subnets, there is

no dispute as to which policy should be applied; it is simply the policy that matches the capabilities of the corresponding client.

TABLE I
ANALYSIS OF IP ADDRESS PROTOCOL USE WITHIN DELEGATED SUBNETS (/24S FOR IPV4 AND /48S FOR IPV6). PERCENTAGES REFLECT THE BOLD HEADING ABOVE. "FAVORED" PROTOCOLS ARE THOSE USED BY MOST CLIENTS.

	IPv4		IPv6	
	Count	%	Count	%
Total Subnets	2^{24}	**100%**	2^{48}	**100%**
No Clients	13,910,051	83%	2.8e14	100%
Some Clients	**2,867,165**	**17%**	**195,036**	**0.0%**
One Client	1,004,614	35%	114,303	59%
Multiple Clients	**1,862,551**	**65%**	**80,733**	**41%**
All UDP	1,543,122	83%	67,065	83%
All Cookie	9,564	0.5%	3,602	4.5%
All TCP	3,659	0.2%	0	0%
Mixed	**306,206**	**16%**	**10,066**	**12%**
Favor UDP	241,031	79%	5,853	58%
Favor Cookies	11,674	3.8%	1,535	15%
Favor TCP[1]	1,531	0.5%	0	0%
No Favor[2]	51,970	17%	2,678	27%

For the subnets for which we observe two or more DNS clients, we analyze the consistency of protocol use. In 84% of multi-client /24 subnets and 88% of multi-client /48 subnets, all IPs exhibit consistent capabilities within the subnet: UDP, Cookie, or TCP. In both cases, an all-UDP subnet is by far the most common, with only a few thousand subnets being multi-clienteted and having all clients using only Cookies or only TCP for their protocol. This leaves just 16% of multi-client /24 subnets and 12% of multi-client /48 subnets that exhibit varying levels of protocol capability.

Overall, the inside of a delegated subnet is fairly uniform as far as DNS transport capabilities. All but 10% of IPv4 /24s with at least one query exhibit a consistent level of transport capabilities. The results are similar for IPv6, with only 4.9% of /48s from which DNS queries were observed having inconsistent transport capabilities. This relatively small group would be required to update their clients before adopting an advertisement record.

C. Estimating Effectiveness for Server Adoption

Advertisement record's effectiveness in reducing reflection-based attacks lies solely in the number of servers that adopt support for enforcing records. If few servers participate, publishing a record will have little to no effect because an attacker can still reflect off of a majority of servers. As noted previously, 58% of /16s produce DNS queries and thus likely host at least one recursive resolver. We expect that accompanying that resolver is an administrator with the knowledge to publish an advertisement record in the DNS. We hope that every subnet publishing a record would also update their servers to perform enforcement.

[1]In subnets where a majority of clients used only TCP for queries, we cannot infer which policy might most appropriately be applied.

[2]In subnets classified as "No Favor", no one protocol was observed more than the others.

We are not able to estimate the reduction in reflection attacks created by a subset of supporting servers because an attacker could simply utilize other servers. In this sense, the protocol may not be effective until a high level of adoption is achieved. However, we do note that attackers prefer to reflect off of servers that generate large responses. If efforts were focused on these servers adopting enforcement, an attack would be weakened and may not be considered cost-effective.

Macfarland et al. found that the average domain produces an amplification ratio of 3 for A records and 6 for ANY records [20]. However, the top 1 million largest domains produced an amplification factor of 20–40. These 1 million domains were served from slightly less than 25,000 (3.7%) of the 670,000 total servers they measured. What this means is that a targeted adoption of this 3.7% of servers would result in the maximum achievable amplification decreasing from $40\times$ to $6\times$—a remarkable improvement. At a $6\times$ amplification factor, the attack against Spamhaus, which created 300Gbps of traffic [21], would have required the attacker to have a bandwidth of 50Gbps as opposed to the estimated 3Gbps. While this value may be achievable by some entities, it significantly raises the cost to participate. A reflection-based attack also loses its value at this bandwidth, as other higher-cost attacks become possible.

Because the response size of recursive resolvers is dictated, in large part, by the responses it receives from the authoritative servers it queries, there is no particular set of recursive servers for which deployment might have a significant decrease in traffic over others. Any recursive resolver is capable of high amplification if the attacker queries them for one of the large domains discussed previously. As a result, all resolvers would need to support advertisement records in order for the full benefit to be apparent.

One final avenue of adoption would be added support by major DNS software such as BIND [22]. If major software added support that worked with a simple configuration option, there may be a substantial uptick in adoption. We note here that DNS Cookies have been supported by major software for up to 5 years at this point, but adoption remains at under 30% [13]. Still, if 30% of servers were to adopt advertisement enforcement, this would theoretically reduce an attacker's volume by 30% if it is assumed that the attacker's pool of servers is representative.

In summary, enforcement by servers is essential, but also the largest challenge faced by the protocol. We hope that as subnets publish an advertisement record, they also update their servers accordingly. We found that if a specific subset of authoritative servers added enforcement, overall amplification would drop significantly. However, the same does not hold for recursive resolvers because of their response size being based on authoritative server response sizes, as discussed previously. We also found that widespread adoption may still be effective as it is non-trivial for an attacker to determine if a server is using enforcement.

D. Server Incentives vs. Costs of Implementing Enforcement

As we have seen, advertisement record's success depends on adoption by authoritative and recursive DNS servers. While these servers play a crucial role, there are not many incentives for them to adopt enforcement. We expect that servers owned by potential victims (i.e., those with advertisement records) would adopt enforcement; however, the majority of servers are likely to be bystanders. Bystanders are used for reflection but do not see a significant gain from enforcing advertisements because they see only a moderate level of traffic during a distributed reflection attack. For example, 30,000 servers were involved in the Spamhaus attack resulting in each only needing to generate $\sim$10 Mbps of traffic [21].

While bystanders have the incentive of adopting enforcement to be good community members, they also face a potentially large burden caused by the increase in the number of queries they would have to perform for checking advertisement records. The scale of this burden depends entirely on the number of unique IP addresses querying the server per day. As we saw in §V-B2, at most 2.3 million IPv4 records would be needed based upon queries sent to the root servers over two days, although the root servers are a special case in many ways, including the clients that query them and the queries that are made [23]. Additionally, the data we used was distributed across 9 server instances.

To gain a better idea of what the cost might look like for deployment with a lower profile, we analyze the recursive and authoritative DNS server logs for Brigham Young University, a mid-size educational institution. For the first week of February 2021, the authoritative servers received unique queries from an average of 15,246 IPv4 addresses per day. The average number of `/16`s and `/24`s from which queries were received per day was 978 and 3,788, respectively. Assuming a typical TTL of one day for an advertisement record, there would be between 978 (i.e., one for each `/16`) and 4,766 (i.e., in the case that every `/24` required a policy delegation: $978 + 3788$) queries needed to lookup policies for every incoming IP. Of course, pinning down the exact number depends on the number of delegations.

We estimate the relative cost for issuing queries by looking at related numbers. First, the authoritative servers we analyzed received an average of 4,298,078 queries per day. Additionally, our recursive resolver averaged 8,617,274 outgoing queries across two 12-hour periods. Given these numbers, the maximum number of outgoing queries (4,766) is less than 0.02% of the total queries sent or received. From this perspective, the load associated with policy lookup seems manageable from both authoritative and recursive perspectives.

We note that these numbers are not representative of other DNS servers. Servers for more popular sites, such as Google or Facebook, likely see far more unique IP addresses. However, these servers are also likely to be designed for a higher volume of traffic.

E. Threat Analysis

The threat model we consider is that of an off-path attacker. The attacker's original goal was to deny service to the victim through a distributed denial-of-service attack. With advertisement records, this is no longer feasible when all servers have deployed enforcement and the victim publishes a record with a secure option (e.g., DNS Cookies). However, an attacker might also try to abuse the advertisement system to (1) deny service to the victim through spoofing an advertisement record or (2) flood enforcing servers with lookup requests. We explore both of these scenarios to gain a better understanding.

1) Spoofing Advertisement Records: One attack vector would be to spoof an advertisement record with a more secure option than is currently in use by the victim. This would effectively deny service to the subnet; DNS servers would reject legitimate packets that are not in compliance with the spoofed record.

The key to mitigating this vector is that the record is published by the owner of the subnet and is queried by servers independently. An off-path attacker would be unable to intercept or modify the query for the advertisement record. Even when considering an on-path attacker, they could spoof an advertisement record to a given DNS server and deny the victim traffic from that server; however, in this position, the attacker could also drop all communication between the two. In other words, the attacker does not gain any advantage when advertisement records are in use.

An off-path attacker could still attempt a cache poisoning attack of the advertisement record for individual servers. If successful, the victim would no longer be able to communicate with the poisoned server which would expect a different protocol than the client uses. This attack would be plausible because an attacker can generally predict when the advertisement query would be sent (shortly after the server receives a spoofed packet). However, cache poisoning attacks can be easily mitigated by the use of DNS Cookies, `0x20` encoding, DNSSEC, or source port randomization by the server. We would expect that servers implementing advertisement enforcement would also deploy one of these methods.

2) Flooding Servers: Another concern may be that an attacker could force a server to query for many advertisement records, causing adverse effects. There are two methods for this attack.

In the first, the attacker could repeatedly query for a subnet with an advertisement record that has a short TTL (e.g., 0). This would require the server to perform a lookup every time a query is received. This attack can be mitigated by the server enforcing a minimum TTL.

In the other attack method, an attacker would send spoofed packets from every subnet to create a large number of needed records for the server. This is especially concerning for IPv6 as there are 2^{40} subnets the attacker could send a query from. This attack is largely mitigated by the server delaying an advertisement lookup until a certain threshold is reached (e.g., performing the lookup on the tenth received query for a subnet).

While both of these attacks have the potential to overwhelm a server, neither of them are unique to advertisement records. Both rely on a large number of inbound queries to the server in the same manner as a typical DoS attack. Compared to a traditional DoS attack that floods the server with unique queries, adding advertisement records would only increase the workload by 10% if the server waited for 10 queries before checking for an advertisement record. We believe that this does not offer a significant advantage to the attacker. Additionally, many servers already employ methods, such as rate-limiting and maximum cache sizes, to mitigate DoS attacks.

There is a potential concern as authoritative servers—which are newly equipped for making queries—may not be robust to these attacks. However, authoritative servers can still use standard techniques to manage inbound queries and can be assisted by a recursive resolver to handle outgoing queries. In essence, following existing best practices should result in servers being able to handle any advertisement-based flooding attacks since these attacks offer only small gains over other methods.

VI. Discussion

A. Limitations and Future Work

Our work has presented a mechanism to enforce secure protocol use by clients and prevent DDoS attacks in the DNS. However, there are several limitations to our work which in turn present future opportunities.

1) IPv6 Considerations: Because there is a significant deployment of IPv6, it is as much part of the attack surface as IPv4. We have attempted to treat IPv6 alongside IPv4 in our analysis. However, we recognize that IPv6 deployment is an ongoing process with current measurements showing less than 30% adoption [24]. As a result, the landscape may change over time and this may affect the practicality of using advertisement records. It is therefore essential that the state of IPv6 deployment be monitored over time to assess the most meaningful deployment of advertisement records.

2) Limited Server Data: One limitation of our work is the minimal dataset we had for the analysis of deployment cost for servers. We only performed an empirical analysis of servers in two categories: the root servers and the servers for our university. Future work should look to analyze other recursive and authoritative DNS server data, including servers across the spectrum of demand. This would allow a more precise estimate of the amount of work required by the server relative to an average load.

3) Accuracy of Server Burden Estimations: Finally, our work was limited in details relating to the exact server implementation for enforcement. For example, we did not specify a precise sequence of steps a server would take when a new query is received. Would the server check its cache for an advertisement policy every time a query is received? What impact on latency would this have? These questions were not answered in our work, but can be explored in the future. There were also multiple instances where we did not define a specific value such as the TTL of a negative record or the number of queries that can be answered before enforcing a policy.

In large part, these specific details were excluded because they have little impact on the overall protocol. However, they must be determined before an implementation can be made. Later work should attempt to implement advertisement enforcement in an existing piece of DNS software (e.g., BIND [22]). This would not only provide a proof-of-concept but would also help in determining specific design choices.

B. Conclusion

In this work, we have presented DNS Protocol Advertisement Records. These records are designed to prevent reflection-based distributed denial-of-service attacks against client subnets who use secure protocols.

With advertisement records, any `/16` (IPv4) or `/40` (IPv6) subnet (and by delegation any `/24` or `/48`) can publicly state which secure transport their clients intend to utilize (e.g., TCP-only or DNS Cookies). DNS servers are then able to look up these records and enforce the provided policy by dropping non-compliant packets. This will effectively prevent reflection attacks because the queries will not be returned to victims.

We evaluated the potential effectiveness of advertisement records. We first saw that adoption for clients is realistic. From an administrative perspective, only a handful of autonomous systems exist per `/16` IPv4 and `/40` IPv6 subnet. We found that on average 57 delegations from `/16` to `/24` would be needed for IPv4 and only 3.4 from `/40` to `/48` in IPv6. Additionally, we saw high uniformity within `/24` and `/48` subnets.

While advertisement records are extremely easy to implement for owners of address space, we saw that there is an increased, albeit manageable, cost for servers to implement the enforcement. Our analysis of the authoritative servers for our university showed that they would need to query for only 4,766 advertisement records—a small amount compared to the volume of queries performed by our institution's recursive resolvers. Despite the manageable cost for servers, they are likely to benefit very little from enforcement. Another key is that the effectiveness of advertisement records depends entirely on adoption by servers. Adoption by all recursive resolvers would ultimately be needed to mitigate attacks (though a portion supporting the protocol may weaken an attack).

Advertisement records have the potential to reduce the effectiveness of denial-of-service attacks that utilize reflection. This will solve a major issue that exists with the current DNS Cookie deployment. The mechanism presented is very easy for clients to adopt, but may not provide enough incentives to see widespread adoption among servers. Since server adoption is key to success, we hope that future work will explore avenues for optimizing the protocol to reduce costs for servers.

Acknowledgments

We gratefully acknowledge DNS-OARC, who provided us access to the 2020 DITL data, the Comcast Innovation Fund for their support of the work that produced this material, and the ICNP 2021 reviewers for their helpful comments.

REFERENCES

[1] M. Prince, "The DDoS that knocked spamhaus offline and how we mitigated it)," 2013. [Online]. Available: https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/

[2] J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, and D. Wessels, "RFC 7766: DNS transport over TCP - implementation requirements," March 2016.

[3] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "RFC 7858: Specification for DNS over transport layer security (TLS)," May 2016.

[4] P. Hoffman and P. McManus, "RFC 8484: DNS queries over HTTPS (DoH)," October 2018.

[5] D. Eastland and M. Andrews, "RFC 7873: Domain name system (DNS) cookies," May 2016.

[6] P. Mockapetris, "RFC 1035: Domain names - implementation and specification," November 1987.

[7] C. Deccio and J. Davis, "DNS privacy in practice and preparation," *CoNEXT 2019 - Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*, December 2019.

[8] S. Kitterman, "RFC 7208: Sender policy framework (SPF) for authorizing use of domains in email, version 1," April 2014.

[9] P. Du and A. Nakao, "DDoS defense deployment with network egress and ingress filtering," *IEEE International Conference on Communications*, July 2010.

[10] M. Jonker, A. Sperotto, R. Van Rijswijk-Deij, R. Sadre, and A. Pras, "Measuring the adoption of DDoS protection services," *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pp. 279–285, November 2016.

[11] J. Postel, "RFC 793: Transmission control protocol," September 1981.

[12] J. Damas, M. Graff, and P. Vixie, "RFC 6891: Extension mechanisms for DNS (EDNS(0))," April 2013.

[13] J. Davis and C. Deccio, "A peek into the DNS cookie jar: An analysis of DNS cookie use," *International Conference on Passive and Active Network Measurement*, March 2021.

[14] P. Vixie and D. Dagon, "RFC draft: Use of bit 0x20 in DNS labels to improve transaction identity," March 2008.

[15] M. V. Larsen and F. Gont, "RFC 6-56: Recommendations for transport-protocol port randomization," January 2011.

[16] M. Andrews, "RFC 2308: Negative caching of dns queries (DNS NCACHE)," March 1998.

[17] F. Denis, "Free ip address to asn database," 2021. [Online]. Available: https://iptoasn.com/

[18] DNS-OARC, "DITL traces and analysis," 2018. [Online]. Available: https://www.dns-oarc.net/oarc/data/ditl

[19] W. Kumari and P. Hoffman, "RFC 8806: Running a root server local to a resolver," June 2020.

[20] D. MacFarland, C. Shue, and A. Kalafut, "The best bang for the byte: Characterizing the potential of DNS amplification attacks," *Computer Networks*, April 2017.

[21] M. Prince, "The DDoS that almost broke the internet," 2013. [Online]. Available: https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/

[22] Internet Systems Consortium, "BIND 9," 2021. [Online]. Available: https://www.isc.org/downloads/bind/

[23] D. Wessels, "Chromium's impact on root dns traffic," September 2020. [Online]. Available: https://blog.verisign.com/domain-names/chromiums-impact-on-root-dns-traffic/

[24] RIPE NCC, "IPv6 enabled networks," 2021. [Online]. Available: http://v6asns.ripe.net/v/6?s=_ALL