A Quantitative Study of the Deployment of DNS Rate Limiting

Casey Deccio
Computer Science Department
Brigham Young University
Provo, UT 84602
Email: casey@byu.edu

Derek Argueta*
Pinterest
San Francisco, CA 94103
Email: dereka@pinterest.com

Jonathan Demke
Computer Science Department
Brigham Young University
Provo, UT 84602
Email: jpd0057@byu.edu

***Abstract*—As a preventative measure against DDoS attacks, many Domain Name System (DNS) software installations include a configurable rate-limiting feature to dismiss abusive traffic. In this paper we conduct a measurement study of the rate-limiting configurations employed by DNS servers for popular domain names, providing a better understanding of the precautions being taken to protect the DNS infrastructure. We believe that an improved understanding of existing defense mechanisms will allow us to propose improvements to the Internet infrastructure that will ultimately yield a more stable and secure internet.**

I. Introduction

The performance and robustness of the Internet is of paramount importance due to the mission-critical applications that rely on it. However, the same high-performance, high-capacity infrastructure used to keep maintain the Internet's stability and resilience can be abused by malicious entities to carry out powerful attacks. Attackers have devised ways to use high-performance Internet hardware to overwhelm unsuspecting victims in a class of attack known as a distributed denial-of-service (DDoS) attack. The attackers issue requests to these Internet servers, claiming to be the victim, so the responses from the servers are directed to the victim, effectively overwhelming it. The powerful Internet servers unwittingly carry out this *reflection* attack with astounding effectiveness, since transmitting and processing requests in the most efficient way possible is their primary objective.

Response rate limiting was developed to combat reflection-based attacks that use Domain Name System (DNS) [1], [2] servers. A knowledge of how this and other defense mechanisms are currently being deployed will serve to both inform Internet researchers on the deployment status of DDoS defense mechanisms and to guide future efforts in combating DDoS. Acquiring this knowledge entails measuring the adoption of DDoS defense mechanisms, characterizing the behaviors associated with their deployment, and measuring their DDoS mitigation impact. With this data in hand, scientists and operators alike can evaluate the effectiveness of these solutions in practice and design and recommend changes to improve DDoS defense.

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

In this paper we present an empirical study of the deployment of DNS response rate limiting across DNS servers associated with popular domain names to measure its deployment and quantitatively assess its impact. We measure key characteristics of the rate limiting configurations, such as rate limit thresholds and reduced response size. Additionally, we evaluate the consistency of rate limiting deployments, across domain names, servers, and protocols, such as IPv6. The primary contributions of our paper are the following:

- A measurement of deployed DNS response rate limiting configurations; and
- An assessment of the DNS response rate limiting deployment consistency.

This work constitutes a more in-depth analysis than treatments in previous studies, and in it we identify behaviors that are potentially problematic, with regard to consistency across domains and network protocols.

II. Background

The Domain Name System (DNS) [1], [2] provides the translation of domain names to Internet resources, most notably Internet Protocol (IP) addresses. This translation process, known as *name resolution*, is the product of communication between many Internet servers. A *stub resolver* issues a query to a *recursive resolver*, e.g., when a Web browser wants to know the IP address for `www.example.com`). The recursive server issues a series of queries to multiple *authoritative servers* in a systematic fashion to find the answer. In addition to returning the response to the requesting stub resolver, the recursive server stores the information in its cache for future queries.

The DNS uses the User Datagram Protocol (UDP) as its primary transport, thus inheriting both the advantages and disadvantages of UDP. A DNS communication involves only a single message in each direction (i.e., a request from the client and a response from the server), so the use of UDP is efficient; no connection setup overhead is required, as it is with the Transmission Control Protocol (TCP). However, UDP lacks the (weak) source identity verification that is inherently part of TCP and its connection establishment. Thus, a DNS server has no way to know whether the source IP address of an

incoming UDP-based request is valid—and that its response is going to the entity that initiated the request.

Leveraging the lack of source validation in UDP, attackers send queries to DNS servers, spoofing the IP address of the victim as the their source, rather than using their own IP addresses. The well-behaving DNS server, unaware of the forgery, sends the responses to the victim, resulting in an behavior known as *reflection*. Because the responses are larger—sometimes hundreds of times larger—than the requests, the attack also exhibits *amplification*. The result of the reflection and amplification carried out by many attackers using many well-provisioned servers is a concentrated flood of network traffic that overwhelms the victims. Even reflectors can be negatively impacted, if they are not as well equipped to handle the load.

DNS response rate limiting [3] is a mechanism that can be deployed by DNS servers to dampen the effects of reflection attacks, servers that would otherwise be desirable reflection accomplices. With response rate limiting, the server observes queries that look alike from a given source IP (presumably a victim of reflection) above a certain rate (the *threshold*). It may then issue responses for only a fraction of subsequent queries and/or return only *truncated* responses. The fraction of queries returned by the servers when the threshold has been reached is referred to as the *slip rate*. Truncated responses are minimally sized and marked as incomplete by the server. A client receiving a truncated response must re-issue the query over TCP, which, as already discussed, has some inherent source verification properties.

III. Previous Work

Best Current Practice (BCP) 38 [4] documents an approach to address the problem at its source; it indicates that routers should drop network packets whose source IP addresses don't originate in the network from which they are arriving. While this solution is seemingly simple, the barrier to its deployment is that the entities required to deploy it are those with the least incentive to do so. The networks hosting attack participants are not negatively affected by the spoofed request traffic. Thus, its deployment remains relatively low [5].

DDoS potential was measured by van Rijswijk-Deij, et al. by analyzing request and response sizes for a large number of DNS domains [6]. Using the observed request and response sizes they calculated the amplification factor and compared it to the theoretical maximum. We also measure response sizes within our own set of domains, but our focus is looking at amplification reduction through the deployment of DNS response rate limiting.

MacFarland, et al. performed two empirical studies of rate limiting on authoritative servers [7], [8]. They used the lack of responses to one or more (five or more in the second study) sequential queries, which sequence included the last query, to indicate rate limiting, with the threshold being the first instance of a query for which no response was received.

Our current work introduces different methodology as well as additional metrics for determining rate limit thresholds, with which we can identify slip rates and other rate limiting behaviors. We also consider IPv4 and IPv6, both to give a more comprehensive analysis and to look for inconsistencies in response rate limiting deployment between the two.

IV. Measuring DNS Rate Limit Configurations

In this section we discuss the methodology we employed to measure the status of DNS response rate limiting. We limited the scope of our analysis to DNS authoritative servers.

A. Domain Name Selection

We used two primary sources to construct the set of authoritative DNS servers we tested for DNS response rate limiting behavior. The first is the DNS root zone file, from which we extracted the top-level domains (TLDs) delegated from the root zone, for which at least one server was responsive. At the time we ran our experiments, the delegations in the root zone totaled 1,386. The second is the list of most popular Web sites, as provided by Statvoo [9], a Web site analysis site. Collectively, the domains we analyzed totaled 922,314. For each of the the root, TLD, and Statvoo domains, we issued queries of type `NS` (name server) to learn the names corresponding to their authoritative servers. For each authoritative server name we issued `A` and `AAAA` DNS queries to learn the corresponding IPv4 and IPv6 addresses, respectively. The domain-server pairs we analyzed totaled 3,872,264.

B. Experiment Methodology

For every domain name in our data set, we analyzed each of its authoritative servers (both IPv4 and IPv6) by issuing various DNS queries. Servers authoritative for multiple domains in our data set were analyzed once for each domain name for which they are authoritative. We issued a burst of 500 queries of type `A` in parallel, all within milliseconds. The intent of the bulk `A` queries is to elicit response rate limiting behaviors of the server, from which we can measure rate limiting thresholds, slip rate, and truncation. We discuss more the nature of our queries, along with the rationale, in Section IV-C.

We developed a custom measurement tool to issue the 500 parallel queries. Our tool was written with the Go programming language to take advantage of its high-performance concurrency; each DNS query was handled as a Go routine. For each query, we recorded the time the query was issued, whether or not a response was received, whether the response was truncated, and the response size, among other useful information.

C. Ethical Considerations

This study involved issuing queries at an abnormally high rate to intentionally invoke defensive behaviors in the form of rate-limiting. However, we wished neither to harm the third-party servers whose behavior we were measuring nor to inadvertently trigger any defense mechanisms that might taint our measurements or even get our activities blacklisted. We therefore took several precautions for the mutual benefit of our study and the servers we analyzed.

We used the A type for our burst of queries, even though it is not the preferred type used by attackers for amplification attacks. While queries for type A generally result in smaller responses, the literature [3] suggests that rate limiting behaviors are generally type agnostic—that is, they are triggered regardless of query type. Thus we felt that by using queries of type A we could get nearly the same results as using another type, but without the negative side effects.

The use of DNS extension mechanisms (EDNS) [10] in DNS queries can (among other things) be used to: 1) request additional (DNS security-related [11], [12], [13]) information from the server; and 2) indicate that the server can exceed the classic maximum response size of 512 bytes. Because our intent was to minimize the size of the responses associated with the burst of 500 queries, we didn't use EDNS in the queries, effectively minimizing the response size associated with the A queries and the possible negative impact to third parties.

We further managed our activities by limiting each measurement burst to a single set of 500 queries within a second. While that disallowed us from learning about rate limiting behaviors whose behaviors required an interval greater than a single second, this allowed us to maintain a relatively low profile. Even with the single burst we were able to infer much from the queries we issued.

Finally, we set up a Web server on the IP address used to conduct our active measurement, serving a Web page that includes information on how to contact us and opt out of our study if desired. We received three inquiries about our rate limiting activities, none of which included a request to opt out.

V. Results

In this section we analyze the results of our experiments, including DNS response rate limit thresholds and other server behaviors. To help us calculate results we first define several terms to describe the queries we issued to each server, as well as their associated responses. Let $Q = \{q_1, q_2, \ldots, q_n\}$ denote the sequence of n queries, each having the same query name and type, issued to a given server within a one-second interval, where q_1 was the first query issued. Let $R = \{r_1, r_2, \ldots, r_m\}$ be the sequence of m responses received from a given server in response to queries in Q, where r_1 was the first response received. We note that because the queries were issued over UDP, neither the order of the arrival of queries nor the order in which responses arrive is deterministic—that is, r_i might not correspond to q_i. Let $R_T \subseteq R$ be the subset of responses for which the TC flag was set, indicating that the response was truncated. Note that we use set operations to represent the association of query and response sequences, such that $Q \cap R$ represents the set of queries for which responses were received.

A. Rate Limiting Thresholds

Among the most important features to extract from the data we collected was the observed rate limiting threshold. This allows us to know which servers are configured to perform rate limiting and how they're configured. Our data contains information from 3,872,264 domain name-authoritative server pairs.

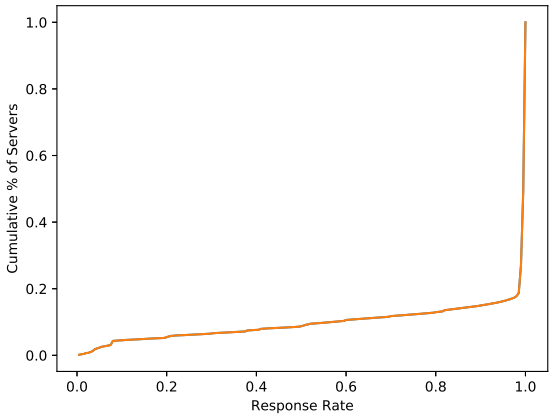

Fig. 1. Cumulative distribution of the overall response rate.

The simplest way to detect the rate limit thresholds exhibited by the servers we queried is to first compare the number of complete (i.e., not truncated) responses returned by the server against the total number of queries issued. This naïve threshold, t, is calculated by considering the response behavior for the entire set of queries, Q, as a collection:

$$t = \frac{|R| - |R_T|}{|Q|}$$

However, there are several potential problems with this naïve approach to rate limit threshold calculation. The fundamental issue is that even servers not configured to rate limit queries are unlikely to respond to 100% of the burst of identical queries that we subjected them to because of the statistical probability of loss due to network or server errors. For example, the plot in Figure 1 shows the comprehensive response rate (i.e.,

$$t = \frac{|R|}{|Q|}$$

) for queries issued to each server for each domain for which it was authoritative. The median response rate was 0.995, indicating at least three responses were dropped by half of servers. As such, a lack of response doesn't necessarily mean that the request was not answered because of rate limiting. This leads to two related side effects. First, the most common rate limit thresholds might not be detected as accurately because they are mixed with the casualties of network loss. Second, the statistical loss results in false positive detection of rate limiting with arbitrary thresholds.

To address these shortcomings, we used a sliding window algorithm to detect the rate limiting threshold for servers queried. Rather than looking only for the first instance of an unanswered query or a truncated response, we iteratively examined sub-sequences (*windows*) of the queries in Q, looking

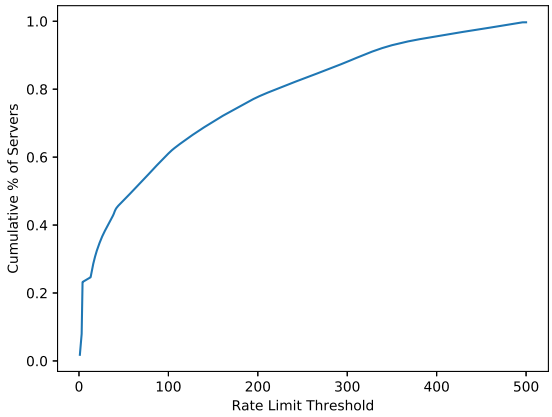

Fig. 2. Distribution of rate limit threshold calculated using the sliding window algorithm with $w = 8$.

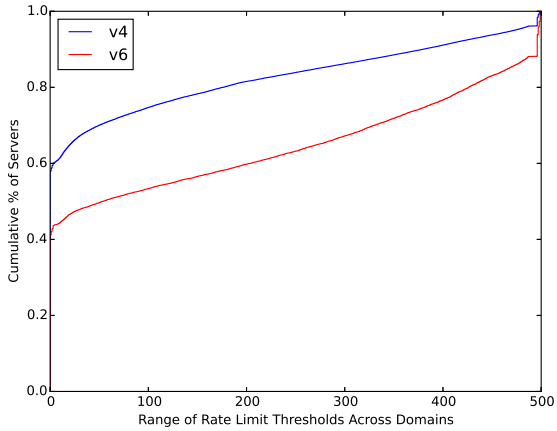

Fig. 3. Cumulative distribution of range of rate limit thresholds exhibited by DNS servers across domains for which they are authoritative.

for behavioral trends within the window that indicate that a rate limiting threshold has been reached by a server. Each window has w elements, $w \geq 1$. Where a window begins at index i (starting with $i = 0$), the query sequence comprising the window is $\{q_{i+1}, q_{i+2}, \ldots, q_{i+w}\}$, such that the chronologically first window considered is: $\{q_1, q_2, \ldots, q_w\}$. Iterating over queries in Q chronologically, the first window in which half or fewer of the w queries received complete (i.e., non-truncated) responses indicated that a rate limiting threshold had been reached. The value of the threshold is the midpoint in this window, i.e., $i + \frac{w}{2}$. Symbolically, this sliding window threshold, $t(w)$, is represented as follows:

$$t(w) = \frac{w}{2} + \min_{0 \ldots n-w} i \mid \frac{|\{q_{i+1}, q_{i+2}, \ldots, q_{i+w}\} \cap (R - R_T)|}{w} \leq 0.5$$

Using a window size, $w = 8$, we identified 645,979 server-domain pairs (16.7% of the total) that were using rate limiting. Figure 2 shows the plot of threshold values as a percentage of servers that were identified as performing rate limiting for their domain.

1) Server Behavior Consistency: We considered rate limiting consistency for the servers that are authoritative for more than one domain in our set. To answer the question of whether a server handled rate limiting consistently among all the domains for which it is authoritative, we calculated the range of thresholds exhibited by a server across all the domains it served. The result is shown in Figure 3 as a cumulative distribution of ranges. The distribution in this graph clearly shows two common themes. While about 60% of IPv4 servers and 45% IPv6 servers behaved with absolute consistency—in terms of rate limit behavior across different domains for which they are authoritative—about 5% of IPv4 servers and 10% of IPv6 servers demonstrated rate limiting extremes for different domains (i.e., the server exhibited rate limiting with a very low threshold for one and didn't rate limit the other at all). The balance of servers exhibited a range of rate limit threshold above 0 and below 490, with the value of these being ranges uniformly distributed.

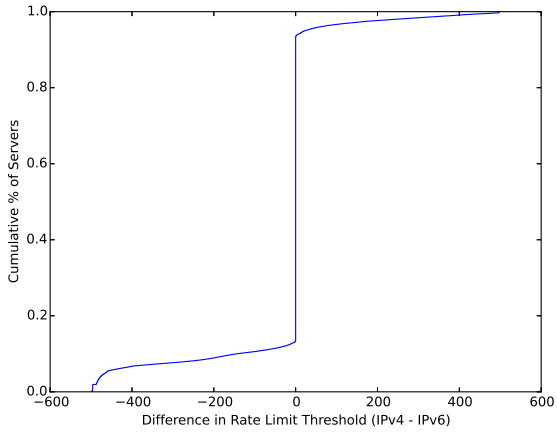

Fig. 4. Cumulative distribution of range of rate limit thresholds exhibited by DNS servers across address families—IPv4 and IPv6.

For a sample of about 462,600 server names (per domain) with both IPv4 and IPv6 addresses, in which rate limiting was detected for at least one address, we analyzed the consistency of rate limiting behavior across these address families. We did this by subtracting the rate limiting threshold measured for the IPv6 address from that measured from the IPv4 address, such that a negative value indicated more aggressive rate limiting (i.e., lower threshold) by the IPv4 address family and a positive value indicated more aggressive rate limiting by the IPv6 address family. Our results are shown in Figure 4. While in about 80% of the cases there was complete consistency between IPv4 and IPv6 addresses corresponding to the same server name, IPv6 had higher thresholds for about 15%, including about 2% of server names for which there was extreme rate limiting on the IPv4 address and no rate limiting at all on IPv6. With the rise in IPv6 deployment, this is another

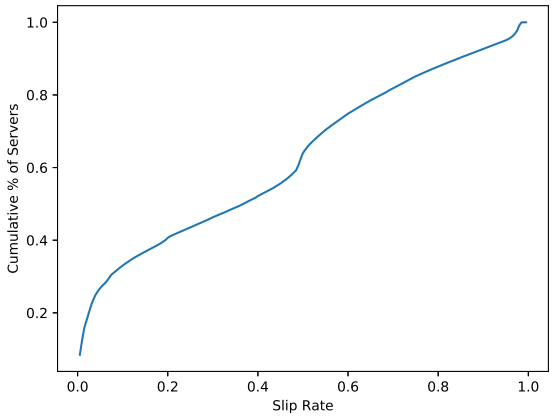

Fig. 5. Distribution of slip rate across rate limiting servers.

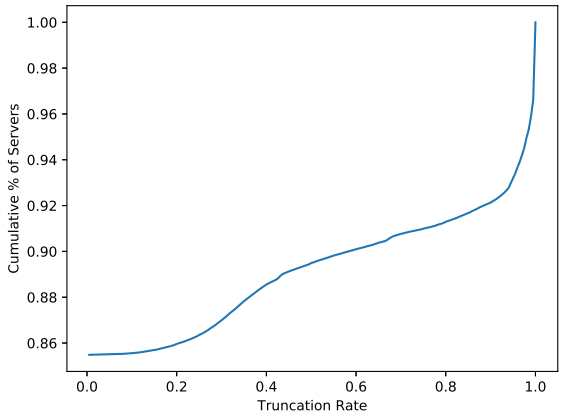

Fig. 6. Distribution of truncation rate across rate limiting servers.

caution that IPv6 configurations should (at least) mirror IPv4, so its neglect doesn't create a loophole for exploitation by malicious entities.

B. Slip Rate

Considering only the servers identified as exhibiting rate limiting behaviors we now examine their slip rate, s—that is, what percentage of responses are returned after the threshold, t, has been reached:

$$s = \frac{|\{q_t, q_{t+1}, \ldots, q_n\} \cap R|}{n - t}$$

Figure 5 shows the distribution of the slip rate among servers exhibiting DNS rate limiting behaviors. The median slip rate for rate limiting servers is 0.4, indicating the half of servers reduce their response rate by at least 60%. One in three rate limiting servers that we analyzed reduced their response rate by 90%. About 5% of rate limiting servers maintained a slip rate of at least 95% after their thresholds had been reached. About 5% of servers employing rate limiting exhibited a slip rate of 0.5. This is not surprising as it is the default value in the literature [3][1].

C. Truncation Rate

Of responses received after the rate limiting threshold has been exceeded, we consider the percentage of responses which are truncated, which we refer to as the *truncation rate*, c, calculated as follows:

$$c = \frac{|\{q_t, q_{t+1}, \ldots, q_n\} \cap R_T|}{|\{q_t, q_{t+1}, \ldots, q_n\} \cap R|}$$

Figure 6 shows the distribution of the truncation rate among servers exhibiting DNS rate limiting behaviors. The plot shows that 85% of servers don't truncate at all. Only about 3% of servers truncate nearly every time. The latter is more like the proposed implementation in the literature [3].

[1]Note, however, that we express slip rate as a fraction, whereas the literature and server documentation often use the inverse, e.g., 2 instead of 0.5

D. Validation

To validate our measurement methodology, we set up a testbed with a Linux server running an instance of the Berkeley Internet Name Domain (BIND) [14] configured as a DNS authoritative server for a contrived domain, which we refer to as `example.com`. From a separate client, we ran our custom rate limiting measurement tool, issuing queries for `example.com` to the testbed server. To emulate some of the uncertainties of the real Internet we configured the server such that:

- DNS responses from the server were dropped with a 1% probability according to a uniform random distribution (using the `iptables` command); and
- A random delay uniformly distributed between 20 and 60ms was introduced for every server response (using the `tc` command).

The random drops and transmission delays introduced at the server simulated DNS response timeouts and out-of-order processing at the client.

In our testbed, we validated our threshold measurement by issuing analysis queries against the BIND server using different configured rate limit thresholds and a fixed slip rate of 0.5. We analyzed the server's behavior with rate limit thresholds of 1 through 10, multiples of 10 between 10 and 100, and multiples of 100 between 100 and 300. We plotted both the naïve threshold and the sliding window ($w = 8$) against the configured threshold in Figure 7. The consistency of our measurements (the Y-axis values) with the configured rate limit settings (X-axis values) is demonstrated by the measurements' close proximity to a line with a slope of 1 (i.e., $y = x$). Both the naïve and the sliding window values are consistent with the rate limit threshold configured at the server.

We validated our slip measurement by issuing analysis queries against the BIND server using different configured slip rates with a fixed threshold of 50. We configured the server with slip values of 0.0, 0.1667, 0.20, 0.25, 0.50, and

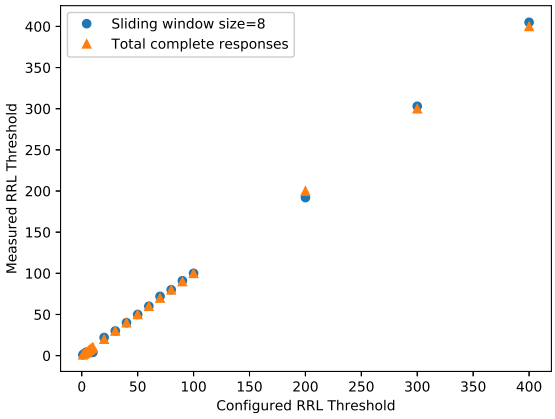

Fig. 7. Measured DNS rate limit thresholds compared against configured rate limit thresholds.

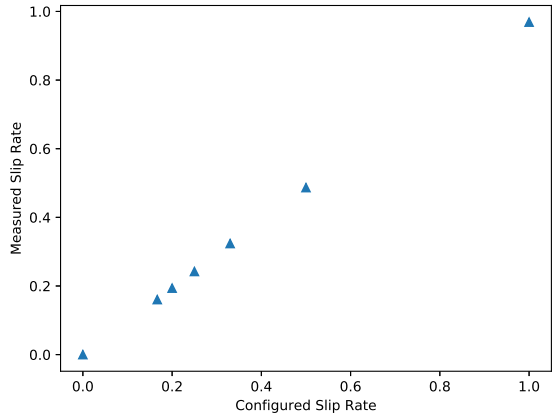

Fig. 8. Measured DNS rate limit slip rates compared against configured slip rates.

1.0. The measured slip values are plotted in Figure 8 against the configured values. Just as with the threshold validation, the line of plotted values having a slope of 1 demonstrates the consistency between the configured and the measured values.

BIND doesn't have a way to configure truncation rate; responses returned after the threshold has been reached are always truncated. We observed that, with the exception of two data points, for which out-of-order, non-truncated responses arrived after the detected rate limit threshold, the truncation rate was 100%.

VI. Conclusion

In this paper we have considered the state of DNS reflection and amplification attacks, reviewed common defense mechanisms, and performed an empirical assessment of their deployment using DNS queries. In particular we have analyzed high-performance Internet DNS servers that are, with their high-performance resources, well positioned to play the role of reflector in reflection-based DDoS attacks. We found that about 16% of authoritative DNS servers employ some sort of rate limiting. We also found that, for the most part, behaviors are consistent across servers for a given domain and for domains common to a server, with some exceptions. However, there is some concern that IPv6 is not being configured as consistently as IPv4 for rate limiting defenses, as demonstrated in our study.

The Internet will continue to be used for both good and for malicious purposes, and finding ways to combat the latter is an important task. Understanding the state of deployment through measurement activities such as these can both inform the present and shape the future, in the defense of the Internet and its users. An important implication of this study and related work is the fact that DNS servers deploying rate limiting and like defensive measures are not the primary targets or victims. Thus, understanding their deployment helps us understand the somewhat altruistic efforts that are being used in defense of the greater good and what might incentivize future such efforts.

References

[1] P. Mockapetris, "RFC 1034: Domain names - concepts and facilities," 1987.

[2] ——, "RFC 1035: Domain names - implementation and specification," 1987.

[3] P. Vixie, "On the Time Value of Security Features in DNS," http://www.circleid.com/posts/20130913_on_the_time_value_of_security_features_in_dns/, September 2013.

[4] P. Ferguson and D. Senie, "BCP 38: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," May 2000.

[5] R. Beverly, A. Berger, Y. Hyun, and k. claffy, "Understanding the Efficacy of Deployed Internet Source Address Validation Filtering," in *Internet Measurement Conference (IMC)*, Nov 2009.

[6] R. A. Sperotto and A. Pras, "DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study," in *Proceedings of the 2014 Conference on Internet Measurement*, ser. IMC '14. New York, NY, USA: ACM, 2014, pp. 449–460. [Online]. Available: http://doi.acm.org/10.1145/2663716.2663731

[7] D. MacFarland, C. Shue, and A. Kalafut, "Characterizing Optimal DNS Amplification Attacks and Effective Mitigation," in *Passive and Active Measurement: 16th International Conference, Proceedings*. Cham: Springer International Publishing, March 2015, pp. 15–27.

[8] ——, "The Best Bang for the Byte: Characterizing the Potential of DNS Amplification Attacks," *Computer Networks (To appear)*.

[9] "Statvoo," https://statvoo.com/top/sites.

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

[11] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "RFC 4033: DNS security introduction and requirements," 2005.

[12] ——, "RFC 4034: Resource records for the DNS security extensions," 2005.

[13] ——, "RFC 4035: Protocol modifications for the DNS security extensions," 2005.

[14] I. S. Consortium, "Berkeley Internet Name Domain (BIND)," https://www.isc.org/downloads/bind/.