THE DNS SINGULARITY: UPTIME BEYOND THE BLACK HOLE

Erjie Zhang, Hezhi Xie, Muhammad Hassnain, Zeerak Babar,
University of California, Davis
{erjzhang, hezxie, mhassnain, zebabar }@ucdavis.edu

ABSTRACT

The Domain Name System (DNS) is a critical component of the internet's infrastructure, translating domain names into IP addresses and thus facilitating user access to online resources. However, its centralized nature makes it vulnerable to Denial-of-Service (DoS) attacks, which can disrupt this essential service and cause widespread accessibility issues. Addressing the vulnerability of DNS to such attacks is vital for maintaining the stability and reliability of the internet.This project introduces a novel distributed DNS framework that significantly enhances resilience against DoS attacks. By implementing a primary-secondary server replication strategy coupled with a load balancer, our system is designed to distribute traffic evenly across the network, thereby mitigating the impact of high-volume traffic on a single point of failure.We evaluated the efficacy of our distributed DNS system by comparing its performance with a traditional single-server DNS setup under a stress test of 50,000 requests in a short time. While the single-server system, running BIND9 with inherent DoS protections, exhibited intermittent downtimes under this load, our distributed model consistently achieved a 100 percent response rate, showcasing its robustness against intense DoS attacks.The results of this study highlight the potential of distributed DNS systems in ensuring uninterrupted internet accessibility, even under aggressive cyber-attack scenarios. This approach not only reinforces the DNS infrastructure but also sets a precedent for future developments in cyber resilience.

1 INTRODUCTION

1.1 INTRODUCTION TO DOMAIN NAME SYSTEM (DNS)

The Domain Name System (DNS) is the backbone of the internet. It translates user-friendly domain names into IP addresses and vice-versa. The DNS infrastructure is made up of computing and communication entities called Name Servers, each of which contains information about a small portion of domain name space. Since the domain name space is organized in the form of a hierarchy, correspondingly there are different types of name servers, including root name servers, top-level domain (TLD) name servers and authoritative name servers. When a user submits a query, the system initially examines the cache to determine if it contains valid information for retrieving IP addresses related to the requested resources. If the cache lacks this information, the system proceeds through a sequence from the root name server to the top-level domain (TLD) name server and finally to the authoritative server in order to resolve the query (Ramasubramanian & Sirer, 2004). Nowadays, DNS is scattered globally and provides numerous internet-related application services. Therefore, the security of DNS is paramount.

1.2 INTRODUCTION TO DENIAL OF SERVICE (DOS) ATTACKS

Denial of Service Attacks perform resource exhaustion on a target, such as a DNS server, by illegally generating large volumes of traffic. This threat can cause huge losses to the security and continuity of a system (Zlomislić et al., 2014). Many aspects of system resources can be abused in these attacks, such as memory and internet connections.

There are multiple ways to perform a DoS attack. First is the Flood Attack. A flood attack is performed by generating a large volume of queries to the DNS server and causing it to be disrupted. The second is the Amplification Attack, which increases the power of the attack by using a reflector. The third is Protocol Vulnerability Exploitation, which takes advantage of the vulnerabilities in the

internet protocol. The fourth is the Malformed Packets focused on security issues in the software implementations. DoS attacks are serious and can cause the server to be unresponsive and even shut down the digital infrastructure completely.

2 Defending Against DoS Attacks

DNS is especially susceptible to DoS attacks due to several security vulnerabilities, including lack of legitimacy verification (there is no mechanism to authenticate the legitimacy of the reply content), open system (the DNS system is open and most DNS servers allow recursive queries), connectionless (DNS uses the User Datagram Protocol (UDP) as its primary method of communication, which does not require a handshake to establish a connection before transmitting data), stateless (the protocol has no memory of transaction) (Fang et al., 2021).

There are various mechanisms for protecting DNS servers from DoS attacks. In general, they can be divided into three types. Nowadays, researchers mainly focus their research on the latter two mechanisms.

- **General mechanisms**: such as disabling unused services, installing the latest security patches, locating DNS servers separately, etc.
- **Tolerance-based mechanisms**: enhance the system's bandwidth (or transit) and server capacity to ensure fault tolerance.
- **Filtering mechanisms**: separate malicious queries from legitimate queries and blocking malicious queries (Zlomislić et al., 2014).

2.1 Tolerance-based Mechanisms

The ultimate objective of DoS attacks is to affect the availability of resources, including network bandwidth, memory resources, and CPU resources. Therefore, in order to mitigate DoS attacks, one way is to enhance bandwidth (or transit) capacity and server capacity.

2.1.1 Enhancing Bandwidth Capacity

Since DoS attacks will cause large volumes of traffic, it is important to make sure the system provides abundant internet connectivity. Therefore, resources should be located close to the end users and large internet exchanges as well. Content Distribution Networks (CDNs) and smart DNS resolution can also help enhance bandwidth capacity by offering services and resolving DNS queries from locations that are closer to the end users.

2.1.2 Enhancing Server Capacity

Another effective strategy to counter DoS attacks is to utilize multiple name servers so that they can quickly scale up on resources to achieve Byzantine Fault Tolerance (BFT), which means the system can continue operating even if one or more of the servers fails. This strategy mainly involves topics related to data replication, data consensus and data security. Besides, other strategies include enhancing the ability of DNS servers to resist DoS attacks by setting longer TTL values (Pappas et al., 2007), using stale cache to cache the stale resources in a DNS server to mitigate the impact of being unable to provide service normally (Ballani & Francis, 2008).

2.2 Filtering Mechanisms

Filtering techniques implement certain rules for detecting and blocking malicious traffic. It basically involves statistical analysis of queries' information such as IP addresses, protocol type, port number, or any other important criteria to find out the malicious queries. For example, history-based source IP filtering aims to classify legitimate traffic by maintaining an IP address database (Zlomislić et al., 2014). There are also other filtering methods such as ingress/egress filtering, router-based packet filtering, capability-based method, Secure overlay Service (SOS), and Source Address Validity Enforcement (SAVE) (Gupta et al., 2012).

3 Replication as a Defense Mechanism

Data replication is an important way to mitigate DoS attacks by enhancing server capacity. It is about replicating data among different servers and distributing DNS queries across these servers. This leads to increased resilience, better load balancing, and lesser response time.

3.1 Replication Techniques in DNS

A zone in DNS is the smallest unit of a replication (Weimer, 2005). The basic data replication method for DNS is called Zone Transfer. It happens between the primary server and the secondary servers, which are all authoritative for that zone. There are two primary ways of zone transfer: Authoritative Transfer (AXFR) and Incremental Transfer (IXFR). AXFR is used to replicate the entire DNS zone file from the primary DNS server to the secondary DNS server (Lewis & Hoenes, 2010). This is typically used when a new secondary server is added to the DNS server, otherwise, it is not very practical to copy the entire zone file when sending the update. In such cases, IXFR is used. IXFR is used to replicate only the changes made to the DNS zone since the last update (Ohta, 1996). This is more efficient than AXFR. AXFR ensures that the primary and secondary servers are in perfect sync. However, it is very resource intensive. IXFR, while being resource-efficient, is more complex to implement.

The relationship between the primary server and the secondary server is shown in Figure 1. The secondary server employs zone transfer to replicate zone data from the primary server, following these steps:

1. The primary server loads zone data from designated zone files;
2. The secondary server loads zone data through a zone transfer operation. Specifically, it periodically examines the Start of Authority (SOA) Resource Record (RR), guided by the refresh parameter within the SOA RR. When it detects that the serial number in the primary server's SOA RR is larger than the current serial number in its zone file, it will initiate a zone transfer query using AXFR or IXFR.
3. Upon any modifications in the primary server's zone file (where each change increments the serial number), the primary server will notify the secondary server through a process named DNS NOTIFY, prompting it to initiate a zone transfer. This ensures that the replicated servers are up-to-date (Chandramouli & Rose, 2009).

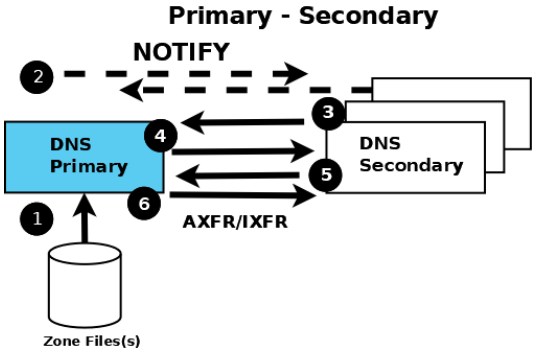


Figure 1: Primary-Secondary Zone Transfer (ISO, 2022)

3.2 Load Balancing Among DNS Servers

In order to enhance the resilience against DoS attacks, it is important to distribute traffic to different DNS servers using load balancing.

There are many methods of load balancing. Anycast is a way of distributing load across various servers. It is a network addressing and routing methodology. It assigns a single IP address to diverse geographic locations. It redirects user DNS queries to the nearest server, thereby reducing latency and load on the primary server. Through this dispersion of traffic, it enhances the system's ability

to withstand a denial-of-service attack. It played a significant role in mitigating the impact of the DDoS attack on the primary DNS server in November 2015 (Moura et al., 2016).

However, Anycast DNS achieves this with the help of the ISP, where they can advertise the same external IP for their multiple servers. It is not feasible for us to get Anycast on board for the implementation of the project.

4 METHODS

Given the challenges associated with implementing Anycast, we devised an alternative plan. Our strategy involves deploying Virtual Machines (VMs) to create a network of DNS servers, including a primary server and multiple secondary servers. Each secondary server maintains identical zone files as the primary server through Zone Transfer. Both the primary server and the secondary servers provide resolution service for the domain name in their zone files.

To fortify our system against potential DoS attacks, we're deploying a load balancer across these DNS servers. The load balancer acts as a gateway for client queries, distributing traffic across the servers. We use Least Connections (LC) algorithm for load balancing, as well as configure proxy response limits and timeout duration for each name server. If one server fails to respond within the specified time or query limit, the load balancer automatically redirects the request to the next available server. This proactive approach ensures uninterrupted service for clients, even if some servers break down due to the DoS attacks.

This multi-server setup significantly bolsters the system's capabilities, surpassing the performance of a single server, and serves as an effective countermeasure against the risks posed by potential DoS attacks.

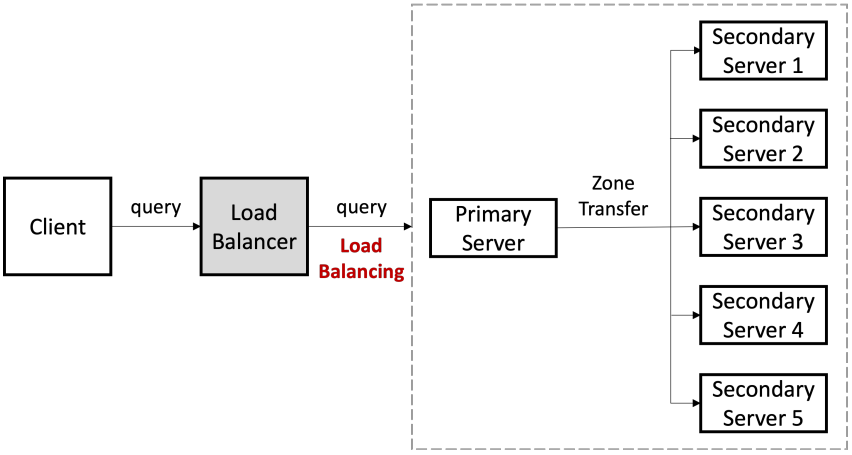


Figure 2: System Architecture

5 EXPERIMENTS AND RESULTS

We use Ubuntu 22 to create and manage 8 VMs, one for the DNS client, one for the load balancer, one for the primary name server, and others for the secondary name server. The IP addresses of these machines are shown in Table 1.

5.1 BUILD DNS SERVERS

The Berkeley Internet Name Domain (BIND) is the most widely used software for implementing DNS servers. It is an open-source software. Therefore, we decide to use BIND9, which is the latest version of BIND to build DNS name servers on these virtual machines.

No.	Role	IP
1	Client	192.168.64.100
2	Load balancer	192.168.64.2
3	Primary name server	192.168.64.10
4	Secondary name server 1	192.168.64.11
5	Secondary name server 2	192.168.64.12
6	Secondary name server 3	192.168.64.13
7	Secondary name server 4	192.168.64.14
8	Secondary name server 5	192.168.64.15

Table 1: IP Addresses of VMs

After installing BIND9 on all of the servers, we configure the primary server as well as the secondary server. The experimental domain name is "**www.example.com**", and the corresponding IP address is "**10.0.0.7**".

We dig the domain name "www.example.com" from the DNS client. The result in Figure 3(a) shows that it has been resolved successfully and replies with the IP address.

In addition to the experimental domain name, the server also works for any normal site. Figure 3(b) is a screenshot for a request to "www.google.com" from the client machine.

(a) dig "www.example.com" (b) dig "www.google.com"

Figure 3: Screenshot of dig command

5.2 Data Replication

We configure the primary server to allow Zone Transfer to all of the secondary servers and also allow notification if there are any zone updates. And we also configure the secondary servers to receive zone transfer from the primary server.

After finishing these configurations, We check the log files of the secondary name servers. They show that the zone data in the primary server has been transferred successfully to the secondary servers. This replication happens automatically periodically so that all the machines stay up to date. Therefore, we have successfully replicated the data among different servers.

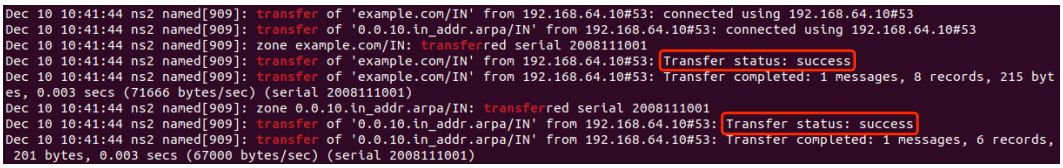


Figure 4: Data Replication by Zone Transfer

5.3 Load Balancing

To implement a load balancer, NGINX serves as our open-source software solution. It is an open-resource software for web serving, reverse proxying, caching, load balancing, media streaming, and more.

The initial step involves configuring the client's name server IP address to match that of the load balancer, thereby directing all client queries to this designated point. Subsequently, we designate all DNS servers as upstream servers for the load balancer.

Upon the foundational setup, we implement the Least Connections (LC) load balancing algorithm. This method ensures that the load balancer directs queries to the server currently maintaining the fewest network connections. Additionally, we set the proxy response limit = 1 and timeout duration = 1s for each server. This configuration dictates that upon receiving a single response, the load balancer promptly halts further waiting, freeing up memory and sockets allocated for that specific session; furthermore, if the load balancer fails to receive a response within this time frame, it proceeds to the next server in the upstream server group, marking the unresponsive server as unavailable for a defined period (which we set as 60 seconds).

By initiating queries from the client and cross-referencing the log files of various servers, we can observe that different servers consistently provide responses as shown in Figure 5. This outcome confirms the successful functionality of the load-balancing system.

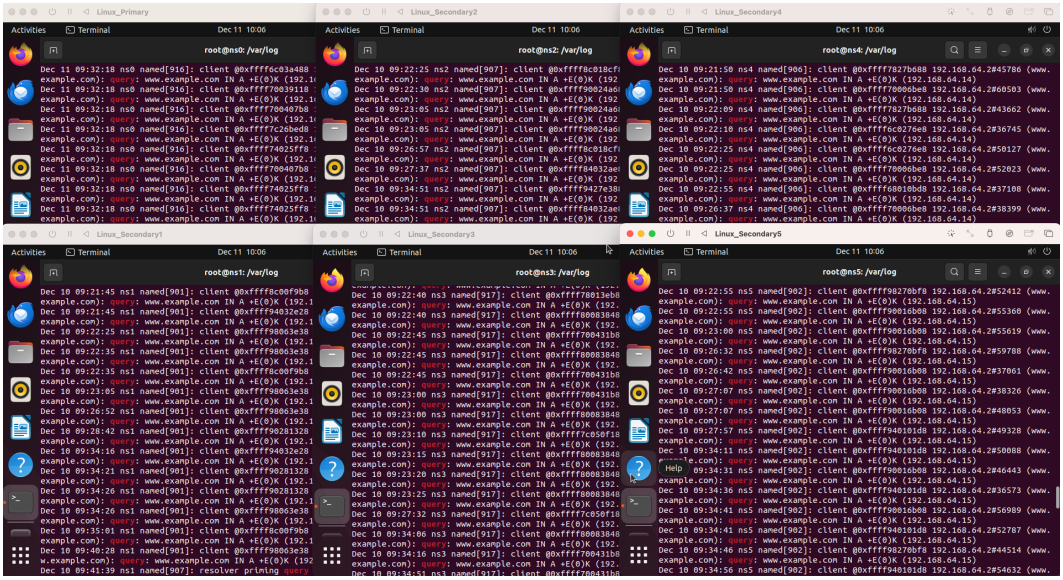


Figure 5: Queries are Distributed to Different Servers

5.4 Evaluation

Due to our limited resources, executing a genuine Denial-of-Service (DoS) attack is unfeasible. Instead, we evaluate our system's performance by simulating a high volume of simultaneous queries from the client and analyzing outcomes across various scenarios:

1. Using a single server;
2. Employing two servers with load balancing;
3. Employing six servers with load balancing.

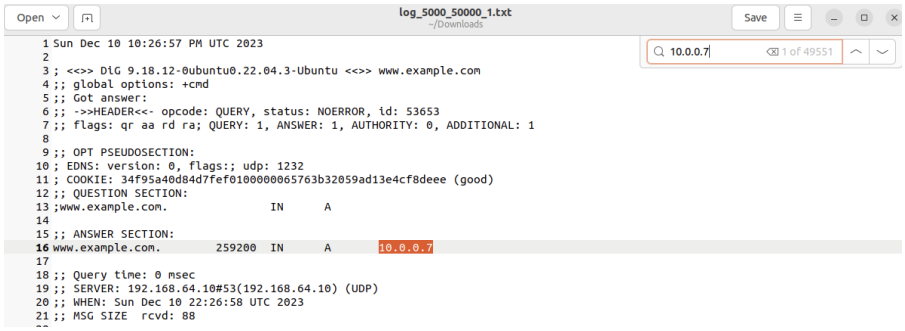
We used a normal DNS Bind9 server as a baseline model to test against. It is to be noted that, BIND9 had built-in DOS protection, so it can help mitigate the risk of abnormal traffic. It is also to be noted that, we were testing this setup on a single machine, so we had technical limitations at place, in terms of launching a denial of service attack. We employed parallel programming techniques to increase

the network traffic, achieving a substantial volume of 50,000 requests in approximately two minutes with a concurrency of 5,000.

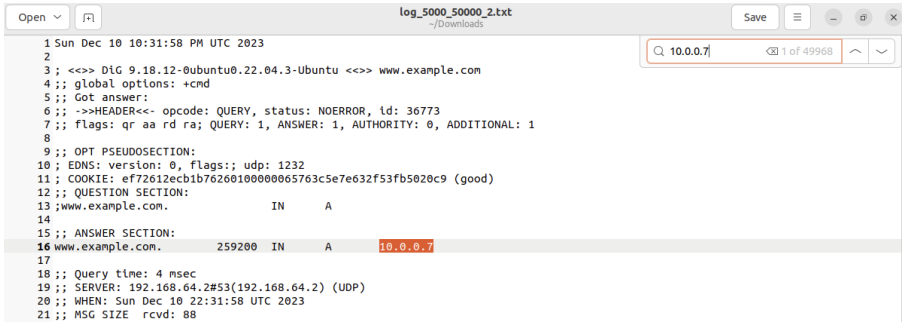
Analysis of the results shown in Table 2 reveals a notable fact: our proposed system successfully responded to queries with zero drop rate, while the single server failed to answer about 0.9% of the network queries.

Scenario	Number of Requests	Number of Answers	Respond Rate
1	50000	49551	99.1%
2	50000	49968	99.9%
3	50000	50000	100.0%

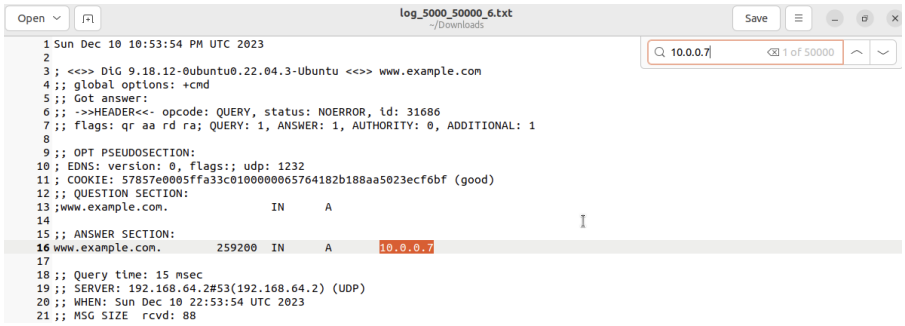
Table 2: Respond Rate Under Different Scenarios



(a) Screenshot of the Result of Single Server



(b) Screenshot of the Result of Two Servers with Load Balancing



(c) Screenshot of the Result of Six Servers with Load Balancing

Figure 6: Screenshot of the Results

6 Related Work

There are various methodologies related to the prevention of Distributed Denial of Service (DDoS) attacks by replicating and load balancing.

Weimer (2005) offered a novel approach by reducing dependence on traditional DNS zone file transfers. It involves capturing DNS queries and responses at network boundaries and extracting relevant data to create a partial replica of the DNS zone. This data is then stored in a database for further analysis. While effective in passively replicating DNS data without active DNS server cooperation, it has shortcomings. Complete DNS zone replication is not guaranteed, leading to potential omissions in DNS records. Additionally, there is a risk of capturing and replicating inaccurate or spoofed DNS data, which may result in erroneous analyses. The replicated data may also be outdated due to its reliance on passive DNS traffic collection. In contrast, our methodology employs a network of Virtual Machine-based DNS servers, including a primary server and multiple secondary servers with identical zone files, ensuring up-to-date and comprehensive DNS record replication.

Park et al. (2004) introduced a cooperative DNS lookup service that aims to enhance the performance of existing nameservers. It utilizes a locality-aware approach to distribute DNS requests, ensuring low-latency and low-overhead name resolution, even during local nameserver delays or failures. Although this method improves DNS availability, it is resource-intensive and presupposes mutual trust among nodes, which may not always be viable, especially if a node is compromised.

Cachin & Samar (2004) focused on distributed domain name service to provide fault tolerance and security, even in scenarios where some servers are corrupted. Key techniques include state-machine replication and threshold cryptography, which secure dynamic updates and protect zone secrets from corrupted servers. However, this method can increase latency in DNS operations and poses challenges in scaling and managing the distributed system. Our approach, by contrast, enhances system resilience and performance by redirecting requests to available servers in case of failure, thus effectively mitigating the risks of DDoS attacks and ensuring continuous service.

7 CONCLUSION

This project presents a significant advancement in the domain of Domain Name System (DNS) resilience, particularly in the context of mitigating Denial of Service (DoS) attacks. Our approach, which revolves around a distributed DNS framework using primary-secondary server replication and an efficient load balancing mechanism, has demonstrated outstanding performance under high stress conditions. The experimental results clearly indicate the superiority of our distributed DNS system over traditional single-server setups. While the single server, even with built-in DoS protections from BIND9, exhibited downtimes under a load of 50,000 requests in a short time, our distributed system maintained a 100% response rate. This underscores the efficacy of our approach in ensuring uninterrupted DNS service, a critical aspect in today's internet-reliant world. The use of virtual machines (VMs) for creating a network of DNS servers, coupled with the implementation of the Least Connections (LC) algorithm and proactive approach for load balancing, proved to be a robust strategy against potential DoS attacks. This setup not only enhanced the overall capacity of the DNS infrastructure but also distributed the traffic load effectively, thus preventing any single point of failure. Our project highlights the potential of distributed DNS systems as a solution to enhance internet security and reliability. It sets a new benchmark for DNS infrastructure design, particularly in scenarios where high availability and resilience against cyber-attacks are paramount.

Future research could explore the integration of advanced machine learning algorithms for predictive traffic analysis and automated threat detection, further strengthening the DNS infrastructure against increasingly sophisticated cyber threats. Additionally, expanding our approach to a real-world, large-scale DNS environment would provide valuable insights into its scalability and adaptability in diverse network conditions.

In conclusion, this project not only demonstrates a practical and effective solution to a critical cybersecurity challenge but also opens new avenues for research and development in the field of DNS security and resilience.

REFERENCES

Hitesh Ballani and Paul Francis. Mitigating dns dos attacks. In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 189–198, 2008.

Christian Cachin and Asad Samar. Secure distributed dns. In *International Conference on Dependable Systems and Networks, 2004*, pp. 423–432. IEEE, 2004.

Ramaswamy Chandramouli and Scott Rose. *Secure Domain Name System (DNS) Deployment Guide:*. US Department of Commerce, National Institute of Standards and Technology, 2009.

Lei Fang, Hongbin Wu, Kexiang Qian, Wenhui Wang, and Longxi Han. A comprehensive analysis of ddos attacks based on dns. In *Journal of Physics: Conference Series*, volume 2024, pp. 012027. IOP Publishing, 2021.

Brij B Gupta, Ramesh Chandra Joshi, and Manoj Misra. Distributed denial of service prevention techniques. *arXiv preprint arXiv:1208.3557*, 2012.

ISO. Bind9 documentation, 2022. URL `https://bind9.readthedocs.io/en/v9.18.20/index.html`.

Edward P. Lewis and Alfred Hoenes. DNS Zone Transfer Protocol (AXFR). RFC 5936, June 2010. URL `https://www.rfc-editor.org/info/rfc5936`.

Giovane CM Moura, Ricardo de O Schmidt, John Heidemann, Wouter B de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. Anycast vs. ddos: Evaluating the november 2015 root dns event. In *Proceedings of the 2016 Internet Measurement Conference*, pp. 255–270, 2016.

Dr. Masataka Ohta. Incremental Zone Transfer in DNS. RFC 1995, August 1996. URL `https://www.rfc-editor.org/info/rfc1995`.

Vasileios Pappas, Dan Massey, and Lixia Zhang. Enhancing dns resilience against denial of service attacks. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pp. 450–459. IEEE, 2007.

KyoungSoo Park, Vivek S Pai, Larry L Peterson, and Zhe Wang. Codns: Improving dns performance and reliability via cooperative lookups. In *OSDI*, volume 4, pp. 14–14, 2004.

Venugopalan Ramasubramanian and Emin Gün Sirer. The design and implementation of a next generation name service for the internet. *ACM SIGCOMM Computer Communication Review*, 34 (4):331–342, 2004.

Florian Weimer. Passive dns replication. In *FIRST conference on computer security incident*, volume 98, pp. 1–14, 2005.

Vinko Zlomislić, Krešimir Fertalj, and Vlado Sruk. Denial of service attacks: an overview. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–6. IEEE, 2014.