DNSBomb: A New Practical-and-Powerful Pulsing DoS Attack Exploiting DNS Queries-and-Responses

Xiang Li[¶], Dashuai Wu[¶], Haixin Duan[¶†‡✉], and Qi Li[¶✉]

[¶]*Tsinghua University*, [†]*Zhongguancun Laboratory*, [‡]*Quan Cheng Laboratory*, ✉*Corresponding Author(s)*

***Abstract*—DNS employs a variety of mechanisms to guarantee availability, protect security, and enhance reliability. In this paper, however, we reveal that these inherent beneficial mechanisms, including timeout, query aggregation, and response fast-returning, can be transformed into malicious attack vectors.**

We propose a new practical and powerful pulsing DoS attack, dubbed the DNSBomb attack. DNSBomb exploits multiple widely-implemented DNS mechanisms to accumulate DNS queries that are sent at a low rate, amplify queries into large-sized responses, and concentrate all DNS responses into a short, high-volume periodic pulsing burst to simultaneously overwhelm target systems. Through an extensive evaluation on 10 mainstream DNS software, 46 public DNS services, and around 1.8M open DNS resolvers, we demonstrate all DNS resolvers could be exploited to conduct more practical-and-powerful DNSBomb attacks than previous pulsing DoS attacks. Small-scale experiments show the peak pulse magnitude can approach 8.7Gb/s and the bandwidth amplification factor could exceed 20,000x. Our controlled attacks cause complete packet loss or service degradation on both stateless and stateful connections (TCP, UDP, and QUIC). In addition, we present effective mitigation solutions with detailed evaluations. We have responsibly reported our findings to all affected vendors, and received acknowledgement from 24 of them, which are patching their software using our solutions, such as BIND, Unbound, PowerDNS, and Knot. 10 CVE-IDs are assigned.

1. Introduction

Denial-of-Service (DoS) attacks [19], prevalent in today's cyber landscape, act by overwhelming target systems or networks and rendering them inaccessible to legitimate users. These attacks have become an insidious threat, with the potential to cripple vital online services and cause substantial financial and reputational damage [18]. Traditional DoS attacks function by continuously flooding a target with superfluous requests and exhausting the available resources. In particular, they often employ reflectors and amplifiers, such as DNS and NTP servers [21], [71]. Such classic hypervolumetric DoS attacks, however, are simple to detect due to continual high volumes of traffic [34], [49].

To make DoS attacks more covert and effective, consequently, Kuzmanovic and Knightly first proposed the Pulsing DoS (PDoS) attack in 2003 [50]. Unlike its continuous counterpart, a PDoS attack is characterized by intermittent traffic bursts aimed at periodically overwhelming the target and a low average traffic rate during the period. This pulsating approach is analogous to a pulse beat, which resonates at the target's weakest points, causing perturbations that can lead to temporary or even permanent system failure. For example, [50] exploits the weakness of the TCP congestion control mechanism, causing packet loss on retransmission and inducing a throughput of nearly zero. Due to the fact that a PDoS attack only generates a series of low-rate sequential packets, its pulsating "on and off" nature enables it to evade detection and is more resource-efficient for the attackers.

Previous Study. After the emergence of the PDoS attack, a number of research studies have been continually devoted to the evolution of this technique through both theoretical analysis and practical attacks (see Section 2). For theoretical analysis, researchers have been implementing the PDoS attack in various networks, such as wireless networks [35], IoT protocols [82], etc., through analyzing specific attack models and simulating with fixed pulsing traffic. For instance, Guirguis et al. [36] point out that any Internet end-systems depending on adaptation mechanisms are vulnerable to the PDoS attack. *All of these works demonstrate that the PDoS attack can effectively cause DoS or service degradation on the target system.* Regarding practical attacks, the security community endeavors to construct practical pulsing traffic using real-world infrastructures at a low sending rate. For example, Rasti et al. [77] first introduce the temporal lensing technique to concentrate DNS queries based on the path latency. Guo et al. [39] further apply the same latency-based method on CDN as the converging lenses. However, *these PDoS attacks either yield a small amplification factor (e.g., 10 of [77]) or require a large pulse period (1,800s of [39]), neither of which are practical and powerful enough to apply.*

Key Observations. In this paper, we observe the capacity of DNS resolvers to concentrate traffic has never been studied in depth. Even though Rasti et al. [77] utilize DNS resolvers to concentrate queries, the path latency limits the accumulating time window to no more than 800ms. Besides, it is challenging to tightly synchronize attack traffic from multiple sources as a bursting pulse at target servers [47], [69]. We find that *the inherent availability-guaranteeing, security-protecting, and reliability-enhancing DNS mechanisms can be exploited to accumulate, amplify, and concentrate attack traffic with a much larger time window and less cost.*

Our Paper. Inspired by the aforementioned key observations, we propose a new practical-and-powerful-ever pulsing DoS attack, dubbed the DNSBOMB attack (see Section 3). With DNSBOMB, attackers could generate a high-volume intermittent pulsing burst in a short period, which is powerful enough to saturate the bandwidth of target networks.

DNSBOMB is made possible through exploiting both DNS queries and responses based on three primary beneficial DNS mechanisms [65], [66], which are widely implemented to guarantee availability, protect security, and enhance reliability (see Section 4). Specifically, first, we employ the availability-guaranteeing DNS mechanism *timeout* to ensure a long period for accumulating queries, e.g., between 1s and 15s, which is greater than the maximum path latency. Second, by leveraging the security-protecting DNS mechanism *query aggregation* on the same domain, we can amplify a small-sized DNS query packet into a large-sized response packet with minimal overload on attackers' nameserver. Third, by delicately delaying the response, we are able to manipulate resolvers to simultaneously return all responses to the spoofed source IP address with the reliability-enhancing DNS mechanism *response fast-returning*. We also provide a theoretical analysis of DNSBOMB to show which metrics influence our attack. For example, a large accumulating time window and a rapid response speed contribute significantly to DNSBOMB. In addition, we introduce additional DNSBOMB attack extensions to increase the attack surface and the impact, e.g., exploiting IP fragmentation [23], [74] and coordinating multiple resolvers like [77].

Experiments and Results. To demonstrate the practical exploiting picture of DNSBOMB, we evaluate 10 mainstream DNS software, 46 public DNS services, and around 1.8M open DNS resolvers (see Section 5). We first calculate the theoretical concentration efficiency, then we measure the real attack impact with controlled experiments, and lastly we present the result with detailed statistics and traffic trend figures. *(i)* By testing locally-installed DNS software, we are able to capture all of their attacking behaviors. *(ii)* For public DNS services, we examine them with ethical considerations, such as only sending 1,000 DNS queries. *(iii)* In order to prevent actual exploitation, we only assess the attack metrics of 1.8M open DNS resolvers, showing their potential power of DNSBOMB. Overall, all resolvers could produce a larger bandwidth amplification factor (BAF) than traditional DoS attacks. The majority of them could result in a much larger BAF than previous PDoS attacks. For example, the pulsing traffic burst (BAF) is *2.9Gb/s* (*21,881x*) for Unbound, while for Yandex DNS it is *876.2Mb/s* (*10,834.0x*).

Furthermore, we conduct controlled real-world experiments to determine how DNSBOMB can DoS or degrade services (see Section 6). Results indicate that DNSBOMB can cause complete packet loss or significant latency for both stateless and stateful services (UDP, TCP, and QUIC).

In conclusion, our experiments demonstrate that *all DNS resolvers can be exploited to conduct more practical-and-powerful* DNSBOMB *attacks than previous PDoS attacks.*

Mitigation and Disclosure. To defend against DNSBOMB, we propose detailed mitigations based on best current practices and assess them with comprehensive experiments (see Section 7), such as reducing the timeout value or extending the response-returning time. Results show *our recommended solutions can effectively eliminate the amplification effect.* For example, the *BAF* of Unbound is reduced by 99.9% to *20.2x*. We have reported discovered findings to all affected parties, including 10 DNS software and 46 public DNS service vendors. We received responses from 39 vendors and discussed mitigations with them, 24 of which have confirmed DNSBOMB, such as BIND, Unbound, PowerDNS, and Knot. 10 CVE numbers have been assigned. We are still awaiting responses from other vendors. We will also publish our evaluation details on https://dnsbomb.net. In summary, our paper calls for a rethinking of the state-of-the-art DNS mechanisms and the mending of potential exploitation risks.

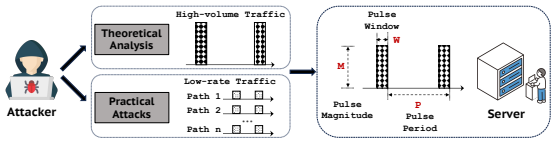


Figure 1. Concept of the Pulsing DoS Attack.

Contributions. We make the following contributions:

- *Comprehensive Survey of Pulsing DoS Attacks.* We provide an in-depth survey of prior PDoS attacks to show the current state and aid in identifying new attacks.
- *Novel Pulsing DoS Attacks.* We propose a new practical and powerful PDoS attack by transforming beneficial DNS mechanisms into malicious attack vectors.
- *Vulnerable Population and Real-world Evaluation.* We conduct extensive experiments to discover the vulnerable resolver population and evaluate the real-world attack impact (an over 20k *BAF* inducing packet loss).
- *Detailed Mitigation and Responsible Disclosure.* We introduce detailed mitigation solutions with evaluation experiments to show the defense effectiveness, and we responsibly report our findings to affected vendors.

2. Current State of Pulsing DoS Attacks

The Pulsing DoS attack (PDoS) is a variant of conventional DoS attacks, which aims to direct short and intermittent bursts of high-volume traffic towards target systems like subsequent and periodic pulses, leading to denial-of-service (DoS) or reduction-of-quality (RoQ). As shown on the right of Figure 1, the PDoS attack differs from traditional traffic flooding attacks that send the attacking traffic continuously by constructing the traffic pulse with a magnitude of M and lasting a time window of W at a period of P.

The Shrew Attack. The PDoS attack was first proposed in 2003 by Kuzmanovic and Knightly [50], named as the shrew attack. The shrew attack attempts to cause packet loss and then degrade the TCP connection by creating short traffic bursts that repeat with a fixed, maliciously chosen, and slow-timescale frequency. It exploits the weakness of the TCP congestion control mechanism that operates on two timescales [26], [72], [75], including Round-Trip Time (RTT) and Retransmission Timeout (RTO). RTT is used

to estimate the optimal amount of data that can be "in-flight" in the network (i.e., data that has been sent but not yet acknowledged) and adjust the sending rate (the congestion window) in normal conditions. If packet loss occurs, TCP will wait for a period of RTO after a packet is resent until receiving a valid packet. Upon further loss, TCP dynamically adjusts RTO based on RTT and its variation and continues the retransmission strategy. If the total DoS traffic during an RTT-length pulse is sufficient enough to induce packet loss, the TCP flow will enter a timeout and resend a packet RTO seconds later. Moreover, if the DoS pulse period approximates the RTO, the TCP flow will continually incur loss as it tries to exit the transmission state, fail to exit, and obtain a nearly zero throughput. Therefore, TCP degradation will occur. Besides, due to the low average traffic bandwidth, such attacks are proved harder to detect than traditional flooding attacks by high-bandwidth traffic analysis.

After PDoS was proposed, numerous studies have been dedicated to investigating this topic. Based on their applicability, we divide them into two categories: theoretical analysis and practical attacks (left-side of Figure 1). Specifically, *we refer to all the low-rate DoS attacks as the PDoS attacks.*

2.1. Theoretical Analysis with Model and Simulation

To demonstrate the PDoS attack impact, Kuzmanovic and Knightly [50] also utilized a combination of analytical modeling, simulations, and Internet experiments to show that their attack could throttle the TCP flows to a small fraction while eluding detection. After that, numerous researchers started to implement the PDoS attack in different networks through analyzing specific attack models and simulating with fixed, precise, and high-volume pulsing traffic.

Initial Stage of PDoS Attack Analysis. Guirguis et al. [36] point out any Internet end-systems depending on adaptation mechanisms are vulnerable to the RoQ attack and exemplify it with a web server under an admission controller. Luo and Chang further analyze the PDoS model [57] and optimize the PDoS attack by maximizing the throughput degradation and minimizing the risk of being detected with a family of objective functions [58]. Ren et al. [78] present the PDoS attack in the mobile ad hoc networks that results in great jitter of goodput and delay. Maciá-Fernández et al. [59], [61] assess the iterative (application) server (e.g., an HTTP server) scenario equipped with FIFO queues and identify significant overload. Zhang et al. [96] study the effect of PDoS attacks on disrupting BGP using fixed traffic pulses (with a pulse magnitude of 185Mb/s and a pulse window of 150ms). They show that BGP sessions are susceptible to PDoS attacks and coordinated attacks can be launched with arbitrarily low-rate individual attack flows.

Development of PDoS Attack Analysis. From 2007 to 2023, numbers of works continue to demonstrate the severe impact of PDoS attacks (DoS or RoQ) on various network scenarios, including dynamic load balancers [37], wireless networks [16], [35], VoIP networks [85], shared links [87], application servers [60], peer-to-peer networks [40], secure channels [41], cloud data center networks [29], server-side sockets [42], feedback-control based Internet services [89], rate-limiting of Xen's hypervisor [90], unsaturated systems [84], cloud auto-scaling mechanisms [10], IoT protocols [82], SDN control channels [15], 4G/LTE networks [28], residential networks [88], low earth orbit satellite networks [30], PDoS-optimizing [94], and RPKI systems [43]. However, they only provide model analysis and experimental simulation instead of any practical attacks.

In conclusion, these works have proven any applications or services containing adaptation or feedback-control mechanisms are susceptible to PDoS attacks. Our PDoS attack is developed based on these works and does not involve model analysis; however, we will present our real-world evaluation.

2.2. Practical Attacks using Real-world Infrastructures

Provided a comprehensive model analysis and simulation of PDoS attacks, the security community attempts to figure out how to construct practical PDoS attacks using real-world infrastructures, i.e., generating multiple low-rate traffic flows and simultaneously directing them towards the target. According to the infrastructure employed by attackers, there are two primary types of attacks: botnet-based and reflector-based practical PDoS attacks.

Botnet-based PDoS Attacks. The straightforward method to generate multiple attack traffic flows is leveraging a collection of "bots" (machines) controlled by attackers. Then, flows can be sent concurrently to the target server. Kang et al. [46] use around thousands of PlanetLab nodes and Looking Glass servers to represent 10^7 bots and concentrate 4Kb/s attack flows into 40Gb/s traffic to flood the specific network link of target servers. They show that their attack could persistently disconnect a target area and is difficult to detect. Shan et al. [83] adopt a centralized strategy with feedback control to coordinate and synchronize bots in an attempt to cause the long-tail latency problem of the target web application. The experiment with 10 machines demonstrates their attacks can achieve the predefined goals.

Reflector-based PDoS Attacks. Real botnets are not easy to obtain and using them would raise ethical concerns [2], [46]; therefore, researchers resort to open Internet reflectors, such as DNS and CDN that contribute to numerous worldwide servers (nodes), for delivering packets. Rasti et al. [77] first introduce the *temporal lensing* technique that concentrates a relatively low-bandwidth flood into a short and high-bandwidth pulse. The key factor is utilizing the *attack path latency*: by first sending packets with longer latencies, and later sending packets with shorter latencies, attackers can schedule different packets rendezvousing at the target server simultaneously within a small pulse window. By leveraging open DNS resolvers as reflectors [71], [81], they discover path latencies from several milliseconds to 800ms for concentrating DNS query packets. In the end, they realize the PDoS attack with a lensing bandwidth gain of around 10 between the pulse magnitude and attacker's maximum sending bandwidth. Bushart [11] further optimizes the temporal

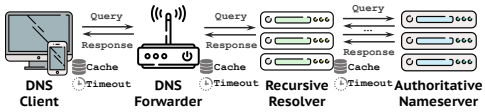


Figure 2. General DNS Resolution Process with Multiple DNS Roles.

lensing technique with DNS CNAME-chaining to increase the amplification factor to 14. Recently, Guo et al. [39] exploit CDN as a converging lens to concentrate low-rate HTTP requests with varying latencies. They present real-world experiments with an amplification factor of 1,527, pulse bandwidth of 872Mb/s, and pulse period of 1,800s.

However, prior studies show it is challenging to tightly synchronize attack traffic from different bots as a bursting pulse at target servers [47], [69], which could reduce the effectiveness of botnet-based PDoS attacks. State-of-the-art reflector-based PDoS attacks either yield a small amplification factor (14) or require a large pulse period (1,800s) that has a negligible impact on normal traffic during that period.

In contrast, our practical PDoS attack, instead of exploiting the attack path latency, could be initiated with an arbitrary pulse period ranging from thousands of milliseconds to any duration of time and can reach an amplification factor greater than tens of thousands of times.

2.3. New Attack Surface in DNS Resolution

The Domain Name System (DNS) operates as a hierarchical and distributed database [65], whose task is mapping human-friendly domain names to machine-understandable IP addresses. The operational mechanics of DNS are built on a client-server model, in which DNS clients initiate queries for domain names and DNS servers return corresponding responses, primarily over UDP [66]. Figure 2 illustrates the general DNS resolution process with multiple DNS roles.

The process starts with a DNS client querying its pre-configured DNS resolvers. Depending on resolution behaviors, there are two types of resolvers: a DNS forwarder that forwards queries to its upstream resolvers, e.g., home or Wi-Fi routers [17]; a recursive resolver, which performs the iterative resolution process per se, such as Cloudflare's `1.1.1.1` and Google's `8.8.8.8`. Specifically, the recursive resolver proceeds with a series of queries to the Root ("."), Top-Level Domain (TLD), and Second-Level Domain (SLD) authoritative nameservers to retrieve the final answer. The returned responses are cached by all "querying roles". Besides, after sending out queries, resolvers will wait a timeout window (e.g., 1s to 15s) for responses. Responses received beyond this time window will be ignored.

Although DNS has been demonstrated to be exploitable to DoS attacks, only the DNS queries or responses are separately capitalized to establish traditional traffic amplification [71], [79] or concentrate pulsing traffic [11], [77]. We discover that the combination of the DNS queries and responses opens long-overlooked attack surfaces to launch new practical-and-powerful-ever PDoS attacks. Specifically, currently prevalent DNS mechanisms provide sufficient time

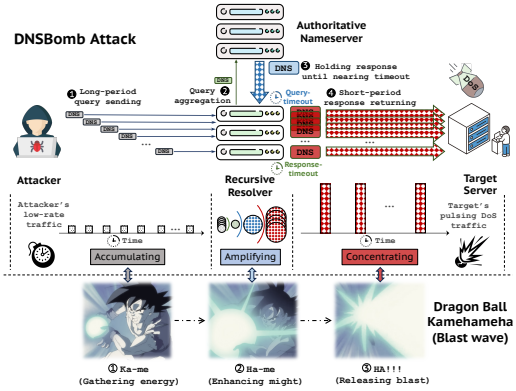


Figure 3. Threat Model of the DNSBomb Attack.

to accumulate DNS queries and enable the rapid packet transmission to concentrate DNS responses with a low cost.

3. DNSBomb Attack Overview

In this paper, we propose a new practical and powerful DNS-based pulsing DoS attack, named as the DNSBomb attack. Rather than leveraging different query path latencies of global DNS resolvers, DNSBomb delicately combines both DNS queries and responses with multiple operational DNS resolution mechanisms and could achieve a bandwidth amplification factor of over *20K*. Inherently, we demonstrate that *the availability-guaranteeing, security-protecting, and reliability-enhancing DNS mechanisms can be exploited to accumulate, amplify, and concentrate attack traffic.*

In this section, we first describe the threat model and then introduce the basic workflow involved in constructing DNSBomb attacks, as well as vulnerable DNS mechanisms. We leave more details and experiments in Section 4 and 5.

3.1. Threat Model

The DNSBomb attack aims to generate short and periodic bursting pulse traffic towards the target server using worldwide open DNS resolvers at a very low traffic cost. The threat model is depicted in Figure 3. Similar to the traditional DNS-based DoS attack, we assume the attacker has the capability of IP spoofing. According to the latest statistics (July 2023) from CAIDA [14], 19.7% of IPv4 ASes and 26.7% IPv6 ASes are identified as IP-spoofable. Attackers could utilize any bulletproof hosting service [1] within these ASes for source IP address spoofing. Besides, the attacker needs to initiate DNS queries for his or her own domain towards exploitable resolvers. Because we primarily use open (public) resolvers as examples in this paper, attackers can send packets directly to them from any location [81]. For resolvers within limited networks, such as private ISP or enterprise networks, attackers can employ internal persons or vantage points like Internet measurement platforms [80] and proxy networks [63]. For the domain, the attacker can purchase any one in the domain registration platform [31], [67] and establish controlled nameservers for management, which will be delicately utilized to conduct this attack.

For the target victims, any server or IP address could be affected. Due to the infamous DNS-over-UDP mechanism, attackers could impersonate any IP as the query's source address and direct the response to that IP. Besides, resolvers exploited to return sizeable response traffic are impacted.

3.2. Attack Workflow

As illustrated in Figure 3, there are three critical steps to construct the DNSBOMB attack: *accumulating DNS queries* (step ①), *amplifying DNS queries into responses* (step ②), and *concentrating DNS responses to the target* (step ③).

The core concept of the DNSBOMB attack is analogous to Goku's signature *Kamehameha* technique (a blast wave attack) in the Dragon Ball series [27]. The Kamehameha is formed when cupped hands are drawn to the user's side to gather energy (step ①: Ka-me), and the energy is enhanced into a large force ball between cupped hands (step ②: Ha-me). The hands are then thrust forward in order to release a streaming, powerful beam of energy (step ③: HA!!!).

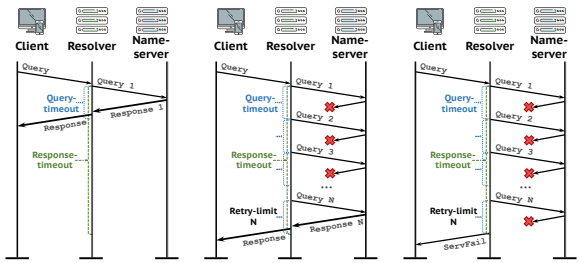
Similarly, DNSBOMB first accumulates sufficient DNS queries to collect bomb energy (Ka-me), then amplifies each query into a larger response for enhancing energy (Ha-me), and finally concentrates those responses into a powerful missile of energy directed at the target server (HA!!!).

Since the attacker can control the accumulating time to collect enough traffic (bomb countdown) and cause concentrated traffic to explode like a bomb, *we refer to our newly proposed PDoS attack as the* DNSBOMB *attack.*

① **Ka-me: Accumulating DNS Queries.** The first phase is accumulating as many DNS queries as possible at a very low rate on the exploitable resolver before response returning (step ❶). To achieve this goal, we employ the availability-guaranteeing DNS mechanism *timeout* to ensure a long period for sending queries. The timeout window, for instance, can range from hundreds of milliseconds to tens of seconds, allowing enough accumulating time. Furthermore, we will show how the *IP defragmentation timeout* mechanism can be integrated to increase the accumulating time.

② **Ha-me: Amplifying DNS Queries into Responses.** The second phase is to amplify a small DNS query packet into a larger response packet (step ❷). To accomplish this, we use our controlled domain (nameserver) and return large-sized responses in accordance with the resolver's capability. Specifically, by exploiting the security-protecting DNS mechanism *query aggregation* on the same domain name, we can significantly reduce our nameserver's overload from receiving tens of thousands of accumulated queries to just a few (or one) queries sent by resolvers. Besides, the nameserver only needs to return one response for the final query to amplify all clients' queries into large-sized responses.

③ **HA!!!: Concentrating DNS Responses.** After accumulating numerous queries and amplifying them into larger responses, in the third phase, by delicately holding the response until nearing timeout in our own nameserver (step ❸), we can manipulate resolvers to simultaneously return all responses corresponding to each query (step ❹). Specifically, because of the reliability-enhancing DNS mechanism *response fast-returning* that the packet should be transmitted as soon as possible, all responses will be concentrated and then sent to the target server in a short amount of time, producing a powerful pulsing DoS traffic.



(a) No Response Delay. (b) Delaying to Timeout. (c) No Response.

Figure 4. DNS Resolution under the DNS Timeout Mechanism.

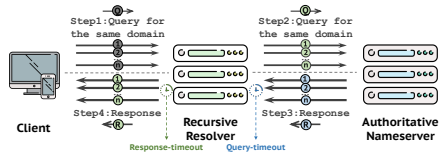
4. Construction of DNSBOMB Attack

In this section, we will describe how to construct the DNSBOMB attack exploiting DNS queries and responses under three steps introduced in Section 3. By leveraging three inherent DNS mechanisms, including *timeout*, *query aggregation*, and *response fast-returning*, we demonstrate that DNSBOMB could produce practical and powerful pulsing DoS traffic (such as over 8.7Gb/s) on target servers with the low-rate attacker-side sending traffic (e.g., only hundreds of Kb/s) and negligible nameserver overload (only observing a few DNS queries and returning one DNS response). The bandwidth amplification factor can be over 20k. Furthermore, we propose additional DNSBOMB attack extensions to strengthen attack impacts and reduce attackers' workload.

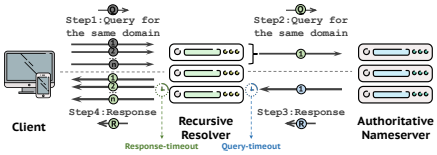
4.1. Accumulating DNS Queries

We utilize the DNS timeout mechanism to accumulate adequate DNS queries for DNSBOMB at a very low rate of delivery, such as hundreds of Kb/s during several seconds. In this part, we first explain the timeout mechanism, then analyze the considerations, and lastly give a detailed method.

Analysis of DNS Timeout Mechanism. Given the stateless nature of UDP [73], typically used by DNS, which lacks inherent delivery promises or retransmission mechanisms, DNS adopts a timeout mechanism to guarantee its availability. The DNS timeout mechanism is an essential operational component within DNS resolution functions [65], including *the query and response timeout* in Figure 4(a). The query timeout designates the maximum duration a resolver will wait for a response from a DNS server before considering the query as timed out. When a resolver sends a query to a nameserver, it will set a timer, such as BIND's default 800ms [9], and wait for the response. If the timer expires without a response being received, the resolver may either resend the query with an updated timer, possibly to a different nameserver, or terminate the request altogether (depicted in Figure 4(b)). Besides, a retransmission limit is typically set to prevent incessant retries, such as the amount of 11

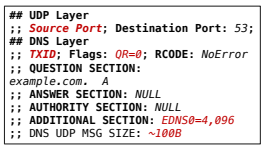


(a) Without the DNS Aggregation Mechanism.

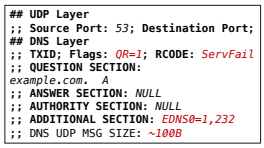


(b) With the DNS Aggregation Mechanism.

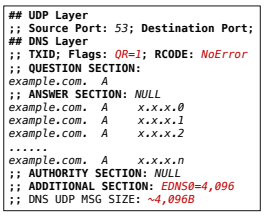
Figure 5. DNS Resolution under Multiple Queries for the Same Domain.



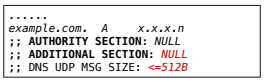
(a) Query with EDNS0.



(b) ServFail Response.



(c) Response with EDNS0.



(d) Response without EDNS0.

Figure 6. DNS Query and Response Packet Examples. Red-signated Fields will be Assigned Different Values Based on the Scenario.

for Unbound [91]. If the response timeout expires (e.g., 10s for BIND [8]) and no response is received, the resolver will return a `ServFail` error answer to the client (displayed in Figure 4(c)). If there is no retry, the query and response timeout are the same. In this paper, *we regard the response timeout as the default timeout value*. This process effectively addresses the risk of indefinite waiting caused by packet loss in the network, ensuring the availability of the DNS system.

Unfortunately, this also opens a new attack surface to construct the PDoS attack. Compared to the classic exploited network path latency, which is at most around 800ms [77], the DNS timeout mechanism provides a much larger window for queries to accumulate, on the order of seconds like 10s.

Accumulating Considerations. When accumulating the real DNS queries, we need to determine that how long the timeout window is and how many queries (*rate-limit*) can be held for a source IP address (the target server). For open source DNS software, we can examine the official documentation or source codes for these configuration values directly. Regarding open DNS resolvers, we have to measure these values beforehand, which will be shown in Section 5.

Accumulating DNS Queries Method. After identifying the timeout and rate-limit values of resolvers, an attacker could accumulate DNS queries towards them. To ensure that each query reaches the resolution timeout limit, attackers employ their own domains. Specifically, for each attack round, attackers should utilize a random subdomain to circumvent the effect of caching and guarantee that the query arrives at the nameserver. On the controlled nameserver's side, the response can then be held until nearing timeout. During the timeout window, sufficient DNS queries should be sent to the resolver at a low rate, one-by-one, until reaching the rate-limit. The sending rate is rate-limit/timeout queries per second (QPS), which is usually hundreds of Kb/s.

4.2. Amplifying DNS Queries into Responses

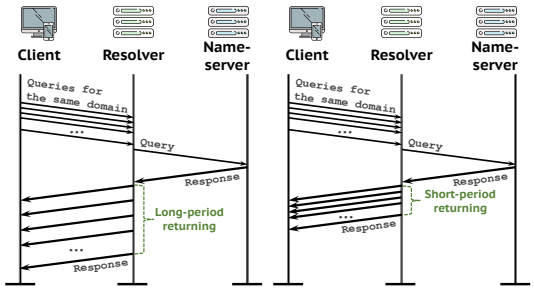
We employ the DNS resolver and controlled nameserver as reflectors to amplify small-sized queries (∼100B) into larger-sized responses (∼4,096B) [71]. However, as shown in Figure 5(a), if the resolver requests the nameserver each time it receives a query (step ❶), the nameserver will be overwhelmed with equal queries and responses (step ❷ ❸).

Here, we show first how to use the DNS query aggregation mechanism to reduce nameserver overloading to only a few DNS queries and one response, then how to enlarge the response, and finally the practical amplifying method.

Analysis of DNS Query Aggregation Mechanism. The DNS query aggregation mechanism was proposed in 2002 to mitigate the birthday attack's effect on DNS servers [44], [68]. The birthday attack is a type of cryptographic attack that exploits the mathematics underlying the birthday problem in probability theory [86]. In the context of DNS, an attacker sends a large number of queries with the same domain in the hopes that one will match a malicious response, leading to cache poisoning. As shown in Figure 5(a), after receiving a query from the client, the resolver will send a new request with different transaction IDs (TXIDs) (step ❷). In response to counteracting this attack, the DNS query aggregation mechanism aggregates identical queries received within a short time window into one single outgoing query, thereby effectively reducing the attack surface. As shown in Figure 5(b), since only one query is delivered to the nameserver (step ❷), the chances of a TXID matching one malicious response significantly decrease. It also reduces network traffic and improves efficiency between the resolver and nameserver. After obtaining a response from the nameserver (step ❸), the resolver will process all queries by returning equal corresponding responses to the client.

Amplifying Considerations. According to RFC 1035 [66], the maximum size of a DNS UDP packet has traditionally been 512 bytes. Considering an average DNS query of 100 bytes, the amplification effect is minimal. To increase the response size further, we leverage the extension mechanism for DNS (EDNS0 defined in RFC 6891 [22]), which allows an extended size of up to 4,096 bytes. Similarly, before amplifying, the attacker needs to determine the maximum DNS UDP size enabled by exploited resolvers and then returns responses of the matching size to them. In Section 5, we will show that, although DNS Flag Day 2020 [24] and resolvers recommend 1,232 bytes of packet size via the EDNS0 option, we can still force some resolvers to return a 4,096-byte response to clients by setting EDNS0 to 4,096.

Amplifying DNS Queries into Responses Method. Here, we use the EDNS0 size of 4,096 as an example. As shown in Figure 6(a), when accumulating DNS queries, we append



(a) Long-period Returning. (b) Short-period Returning.

Figure 7. DNS Response Returning Examples of Resolvers.

each with an EDNS0 option and set its value to 4,096. Then we return a response with as much data as needed to amplify the packet to 4,096B. If the packet size exceeds the enabled EDNS0 size like 1,232B, the resolver will return a `ServFail` response illustrated in Figure 6(b). Otherwise, the resolver will return the entire 4,096-byte response in Figure 6(c) via IP fragmentation to the client. For a query without EDNS0, the response packet in Figure 6(d) is no larger than 512B.

4.3. Concentrating DNS Responses

The final stage is to concentrate all accumulated-and-amplified DNS responses and direct them as a short pulsing traffic to the target. This part will first describe how we utilize the DNS response fast-returning mechanism to concentrate packets into a short pulse window (such as tens or hundreds of milliseconds) and generate a traffic pulse of over 1Gb/s, followed by practical considerations and methods.

Analysis of DNS Response Fast-returning Mechanism. The DNS response fast-returning mechanism is a strategy employed by DNS resolvers in lieu of a standard DNS mechanism to expedite the resolution process and enhance reliability. Essentially, it requires the resolver to send back the response to the client as soon as it receives a valid response from the upstream servers. In this manner, the resolver can significantly reduce the latency experienced by the client. Since it is not an universally adopted standard, its implementation may vary among different DNS implementations. Several resolvers may return the responses slowly, as shown in Figure 7(a). Nonetheless, in Section 5, we will demonstrate that the majority of DNS implementations [5], [76] and services return hundreds of responses in just a few milliseconds, such as 50ms in Figure 7(b). This enables the attacker to concentrate maliciously crafted responses into a traffic burst with a pulse window of tens of milliseconds.

Concentrating Considerations. To optimise the concentrating effect, the attacker needs to guarantee adequate numbers of responses are returned simultaneously. First, when accumulating queries, each query should be allocated a distinct source port or TXID in order to be distinguished from the others as shown in Figure 6(a). Then all of them will receive a response. Second, the attacker should examine the response-returning speed in advance and select the faster ones. Finally, by forging the source IP address, the attacker could direct all pulsing traffic to the target server.

Concentrating DNS Responses Method. When all queries are accumulated and prepared to receive larger responses with EDNS0, although the resolver will retry several times, *attackers only need to return one delicately crafted large-sized response for the final query from the nameserver just before the timeout.* The response size should not exceed the maximum packet size enabled by exploited resolvers. Then, all responses will be transmitted quickly to the target server, for example, which can produce powerful pulsing DoS traffic with a pulse magnitude of more than 1Gb/s.

4.4. DNSBomb Attack Metrics

In this part, we provide a theoretical analysis of the DNSBomb attack with the following annotations and equations to show which metrics have an impact on our attack.

The metrics for the attacker's and nameserver's side:

- P_A: # of accumulated DNS query packets.
- S_A: Packet size of accumulated DNS queries.
- T_A: Time used when accumulating queries.
- P_{NQ}: # of DNS queries sent to the nameserver.
- P_{NR}: # of DNS responses returned by the nameserver.

The metrics for the victim's side:

- P_V: # of concentrated DNS response packets.
- S_V: Packet size of concentrated DNS responses.
- T_V: Time used to return concentrated responses.

The metrics of concentration efficiency:

- *PCE* (*packet number concentration efficiency*): $\frac{P_V}{P_A}$
- *SCE* (*packet size concentration efficiency*): $\frac{S_V}{S_A}$
- *TCE* (*time concentration efficiency*): $\frac{T_A}{T_V}$
- *BCE* (*bandwidth concentration efficiency*): $\frac{P_V}{P_A} * \frac{S_V}{S_A} * \frac{T_A}{T_V}$

BCE is also referred to *BAF* (bandwidth amplification factor). As shown by these concentration efficiency metrics, the more the concentrated DNS responses are (the larger the response packet size and the shorter the concentration time), the higher the concentration efficiency will be.

With regard to the nameserver, P_{NQ} is at most the query retry-limit value, same to P_{NR}. In practice, P_{NQ} is typically a small number, such as 10, and the attacker only needs to return one response for the final retrying query ($P_{NR} = 1$), which has negligible workload on the nameserver.

4.5. DNSBomb Attack Extensions

Here, we propose additional DNSBomb attack extensions to further enlarge the attack surface, enhance the attack effect, and reduce the attacker's workload.

Exploiting a DNS Forwarder. In Figure 3, we only show how to exploit a recursive resolver to construct DNSBomb. However, a DNS forwarder is vulnerable to DNSBomb as well. We can regard the forwarder as the ingress while its upstream server as the egress [81] and use the smaller rate-limit, max-packet-size, and timeout values between them to conduct our attack similar to that of the recursive resolver.

Increasing the DNS Query Accumulating Time with IP Fragmentation. Although the DNS timeout mechanism

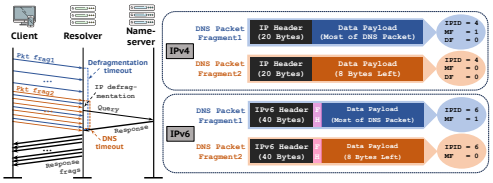


Figure 8. DNS Resolution under the DNS Timeout and IP Defragmentation Timeout Mechanism.

has provided sufficient time, such as 10s, to accumulate queries, we find the IP defragmentation timeout can further prolong this time window. IP fragmentation is used to divide packets into smaller fragments for transmission [23], [74]. Each fragment of the original packet carries an `IPID` field and a `MF` flag to indicate more fragments or not. Received fragments are stored in the kernel's defragmentation buffer for a timeout pending reassembly. For example, Linux's default defragmentation timeout for IPv4 is 30s and for IPv6 is 60s [56]. The maximum number of fragments that can be held in Linux's reassembly buffer is 64 for IPv4, while unlimited for IPv6. As shown in Figure 8, we can split a query packet into two fragments: $fragment_1$ containing the majority of the packet and $fragment_2$ with left 8-byte payload (because the fragment data payload must be at least 8 bytes long [23], [74]). All the $fragment_1$ packets are sent to the resolver during the defragmentation timeout, while the $fragment_2$ packets are delivered within the DNS timeout before the defragmentation timeout. When a DNS query is reassembled, resolvers will request the nameserver. The nameserver returns the response until nearing timeout, assuring a large accumulating time window (30s or 60s).

Attacking the Shared Link prior to the Target Server. As shown in Section 4.1, resolvers employ a rate limit strategy for each IP address. For example, the default rate-limit for Google Public DNS is 1,500 QPS [33]. To circumvent this limit, attackers could spoof multiple IP addresses that share the same upstream router node (link) and concentrate all the rate-limit-pack of responses to that shared link like [46]. Specifically, for each IP address, due to rate-limiting, the resolver can only return a maximum of *rate-limit* responses. If the rate-limit value is small, the attack traffic might be minimal. However, if the attacker spoofs multiple source IP addresses sharing a same upper link, all these limited attack traffic could be concentrated to that link simultaneously, thus overwhelming that link and all downstream servers.

Coordinating Multiple DNS Resolvers. Although a single resolver can generate a large pulsing traffic of hundreds or thousands of Mb/s, an attacker could also utilize multiple resolvers to further enhance the DNSBOMB attack like prior PDoS attacks [77]. To coordinate all single-pulsing traffic into a larger one, attackers must measure the network path latency in advance and make them arrive at the target simultaneously [69]. Specifically, to measure the latency between the target and resolver, we can use the King method [38] that could approximate the latency between arbitrary Internet hosts. However, we acknowledge that time synchronization makes it difficult to coordinate multiple resolvers [69]. In addition, by exploiting resolvers one by one or employing a small accumulating window, such as 50ms, the attacker can manipulate them to produce continuous DoS traffic that arrives sequentially, similar to traditional DoS attacks [79].

5. Vulnerable Resolver Population

In this section, we attempt to demonstrate the practical exploiting effect of DNSBOMB by assessing 10 mainstream DNS software, 46 public DNS services, and 1.8M open DNS resolvers. Based on the construction steps in Section 4, we first examine their attack factors, including *rate-limit*, *timeout*, *query count*, *default EDNS0*, *maximum DNS packet size* w/o EDNS0 options, and *1k-response-returning time*, to calculate the theoretical concentration efficiency. Then we conduct real experiments to measure their practical attack impacts with ethical considerations. In the end, we show that all resolvers can produce a larger *BAF* than traditional DoS attacks. In addition, the majority of them could cause a much larger *BAF* than prior PDoS attacks, such as Unbound's pulsing traffic of *2.9Gb/s* and *BAF* of *21,881x*.

Experiment Design. To investigate these factors, we first inspect their official documentations and websites and source code, for publicly available information. Then we perform "blackbox-testing" by registering our own domain and configuring a nameserver to return delicately crafted responses. For example, when testing the timeout and packet size, we hold the response for several seconds or reply with responses of different size to determine whether we can receive a valid response. The query count can be obtained by collecting the request sent to our nameserver. For the left factors like rate-limit, EDNS0, and response-returning time, we can analyze them by capturing the traffic on the client side. Finally, we launch small-scale experiments to assess their attack effects in controlled environments. According to Google's query rate-limit of 1,500 [33], we select 1,000 as the number of queries to reduce the overload on resolvers and targets.

5.1. DNS Software

DNS Software List. We select 10 mainstream DNS software that are also used by previous work [52], [54], [55], [93], [95], as shown in Table 1, two of which are DNS forwarders, including Dnsmasq and CoreDNS.

Experiment Setup. We install their latest versions on machines running Ubuntu 22.04 or Windows Server 2022, respectively, and use another Ubuntu 22.04 host as the client to send queries. The forwarder's upstream server is pointed directly to our nameserver, while all the other software run in the recursive mode. All machines are linked to a local network with a 10Gb/s network bandwidth.

Theoretical Analysis. After investigation and experiments, we list the results of tested factors in Table 1. Specifically, software other than BIND, PowerDNS, MaraDNS, and Dnsmasq enables a rate-limit of over 3k pps. All the timeout values are above 1.5s, which is significantly greater than the network path latency of 800ms and provides sufficient time for accumulating queries. BIND and Unbound even have a default timeout of 10s. The query count is below 10,

TABLE 1. DNS RESOLUTION MECHANISMS RELATED TO DNSBOMB AND THEORETICAL ANALYSIS OF 10 DNS SOFTWARE.

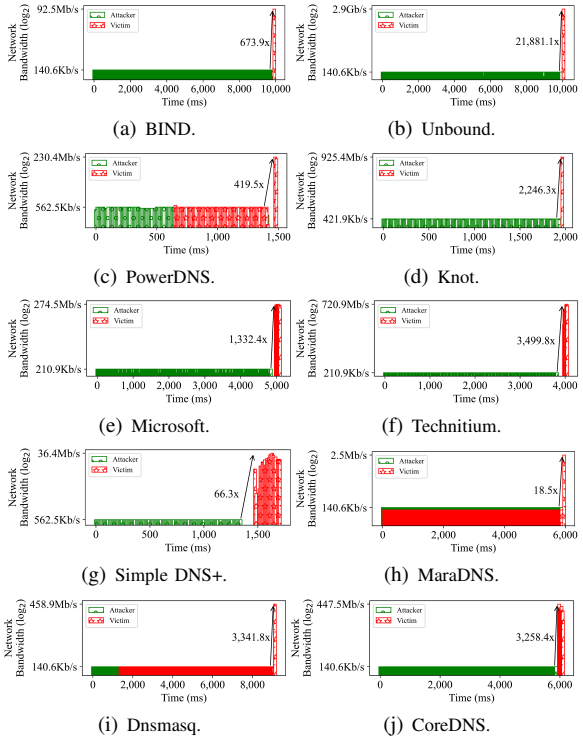
DNS Software		Accumulating		Amplifying				Concentrating	Theoretical Analysis			
Brand	Version	Rate-limit (#)	Time-out (s)	Query (#)*	EDNS0 (B)	Max. Size (B) O. E.[1]	Max. Size (B) W. E.[2]	Returning Time (ms)[3]	*PCE*[4]	*SCE*[5]	*TCE*[6]	*BCE*
BIND	9.18.20	130	10	6	1,232	512	1,232	8	1x	12.3x	1,000x	12,320.0x
Unbound	1.19.0	20k	10	9	1,232	512	4,096	10	1x	41.0x	1,000x	40,960.0x
PowerDNS	5.0.0	500	1.5	1	512	512	1,232	20	1x	12.3x	75x	924.0x
Knot	5.7.0	30k	2	4	1,232	512	1,232	58.7	1x	12.3x	34.1x	419.8x
Microsoft	2022	30k	5	1	4,000	512	4,000	128	1x	40.0x	39.1x	1,562.5x
Technitium	11.0.2	30k	4	3	1,232	512	4,096	36.5	1x	41.0x	109.6x	4,488.8x
Simple DNS+	9.1.116	3k	1.5	3	1,280	512	1,280	240	1x	12.8x	6.3x	80.0x
MaraDNS	3.5.0036	7	6	6	-	512	512	0.5	1x	5.1x	600x	3,072.0x
Dnsmasq	2.89	150	∞[7]	1	4,096	2,300	4,096	0.5	1x	41.0x	1,000x	40,960.0x
CoreDNS	1.10.1	3k	6	2	4,096	512	4,096	140	1x	41.0x	42.9x	1,755.4x

[1]: Max. DNS packet size enabled for queries without EDNS0. [2]: Max. DNS packet size enabled for queries with EDNS0=4,096.
[3]: Response returning time for 1,000 packets. [4]: Number of reply/Number of query. [5]: Maximum DNS packet size/100.
[6]: Timeout/Response-returning-time (Time less than 10ms is considered as 10ms). [7]: Unlimited (Treated as 10s).
*: The number of queries sent to the nameserver if not receiving responses during the timeout window.

TABLE 2. DNSBOMB EXPERIMENT RESULTS OF 10 DNS SOFTWARE USING 1,000 DNS QUERIES.

Software	Practical Attack Bandwidth			
	Attacker-side	Victim-side	Nameserver-side	*BAF*
BIND	140.6Kb/s	92.5Mb/s	155.5Kb/s	673.9x
Unbound	140.6Kb/s	2.9Gb/s	140.6Kb/s	21,881.1x
PowerDNS	562.5Kb/s	230.4Mb/s	70.3Kb/s	419.5x
Knot	421.9Kb/s	925.4Mb/s	70.3Kb/s	2,246.3x
Microsoft	210.9Kb/s	274.5Mb/s	70.3Kb/s	1,332.4x
Technitium	210.9Kb/s	720.9Mb/s	140.6Kb/s	3,499.8x
Simple DNS+	562.5Kb/s	36.4Mb/s	1,167.4Kb/s*	66.3x
MaraDNS	140.6Kb/s	2.5Mb/s	123.4Kb/s	18.5x
Dnsmasq	140.6Kb/s	458.9Mb/s	210.9Kb/s	3,341.8x
CoreDNS	140.6Kb/s	447.5Mb/s	468.0Kb/s*	3,258.4x

1. The query sending time and response packet size are set to the maximum values listed in Table 1.
2. The nameserver's receiving bandwidth is $\leq$ attacker-side's since resolvers retry < 10 times except for * (no query aggregation).
3. The nameserver's sending bandwidth can be negligible since it only needs to return one response for the final query.



(a) BIND. (b) Unbound. (c) PowerDNS. (d) Knot. (e) Microsoft. (f) Technitium. (g) Simple DNS+. (h) MaraDNS. (i) Dnsmasq. (j) CoreDNS.

Figure 9. DNSBOMB Experiment Network Bandwidth of 10 DNS Software using 1,000 DNS Queries.

indicating that *these resolvers, despite receiving thousands of queries on a same domain name, request the nameserver for responses no more than 10 times due to query aggregation.* This substantially reduces the nameserver's workload. Simple DNS+ and CoreDNS do not employ the query aggregation mechanism. Although the default packet size is 512B, with the EDNS0=4,096 option, the packet size can be 1,232 or even 4,096 bytes (such as Unbound and Technitium), which violates the DNS Flag Day 2020 [24]. Besides, all software can send 1k responses within 250ms or even 50ms like Unbound and Knot. In conclusion, theoretically, nearly all software could produce powerful DNSBOMB attacks with a *BAF* in the thousands or even tens of thousands.

Practical Attacks. The practical attack bandwidth results are listed in Table 2. All *BAF* numbers are greater than the traditional packet size-based amplification attacks (*SCE*), demonstrating the effectiveness of response concentrating. The attacker-side bandwidth (that *we use a 10ms time window to calculate the network bandwidth*) is only hundreds of Kb/s, while the victim-side bandwidth is nearly hundreds of Mb/s or even several Gb/s. The nameserver's bandwidth is negligible because: *(i)* due to query aggregation, only less than 10 queries are sent to it; *(ii)* it only needs to return one response for the final query that triggers the resolver to return thousands of responses to the target server simultaneously (explained in Section 4.2 and 4.3).

Notably, the practical *BAF* listed in Table 2 is less than the theoretical *BCE* shown in Table 1. As analyzed in Section 4.4, *BAF* (*BCE*) is equal to $PCE * SCE * TCE$. Any factors that potentially decrease *PCE*, *SCE*, or *TCE* could influence the attack result. When calculating the theoretical

TABLE 3. DNSBOMB EXPERIMENT RESULTS OF UNBOUND WITH DIFFERENT TIMEOUT USING 1,000 DNS QUERIES.

Timeout (s)	Practical Attack Bandwidth		
	Attacker-side	Victim-side	*BAF*
1	843.8Kb/s	2.8Gb/s	3,480.9x
2	421.9Kb/s	3.0Gb/s	7,354.0x
3	281.2Kb/s	3.0Gb/s	11,313.9x
4	210.9Kb/s	2.9Gb/s	14,240.4x
5	210.9Kb/s	2.6Gb/s	12,716.8x
6-10	140.6Kb/s	3.0Gb/s	22,627.8x

The response packet size is set to 4,096B.

TABLE 4. DNSBOMB EXPERIMENT BANDWIDTH (GB/S) OF UNBOUND WITH DIFFERENT NUMBER OF INSTANCES AND QUERIES.

# of Unbound	# of DNS Queries									
	1k	2k	3k	4k	5k	6k	7k	8k	9k	10k
1	3.0	3.0	2.9	3.7	3.5	2.6	2.1	3.6	2.2	3.4
2	2.6	5.5	3.2	4.3	2.9	4.7	6.7	6.2	4.4	6.0
3	4.6	6.2	4.8	5.6	2.4	6.8	4.7	8.7	3.9	3.2
4	4.9	4.3	7.5	2.5	4.8	5.0	3.5	3.3	4.5	5.2
5	2.8	3.7	4.5	4.8	3.8	4.5	4.6	3.6	2.7	3.3
6	3.1	7.5	5.1	6.8	7.4	2.6	6.2	6.6	4.6	5.4
7	6.9	4.4	2.2	2.7	1.9	5.6	2.9	2.3	2.3	6.6
8	1.4	7.4	4.3	5.5	3.2	3.3	2.1	3.9	2.3	8.7
9	5.0	4.4	2.5	2.5	5.2	2.7	2.5	4.6	3.3	5.0
10	2.5	2.3	3.4	3.3	6.7	7.1	4.0	3.2	3.2	3.3

The response packet size is set to 4,096B, while the timeout to 10s.

BAF, we assume that *(i)* each query gains its corresponding response (*PCE*); *(ii)* every returned response is amplified to its maximum enabled size (*SCE*); *(iii)* all the responses are transmitted to the target within a short period (*TCE*). Then all efficiency values can reach the number in Table 1, which induces a large theoretical *BAF*. However, in practical experiments, software show different behaviors that reduce the efficiency values. For example, Dnsmasq restricts the number of acceptable queries, while Simple DNS+ returns responses slowly. Thus, since we send too many queries that are impacted by these settings, the practical *BAF* is small.

In detail, Unbound shows the highest *BAF* of *21,881.1x* and largest pulse magnitude of *2.9Gb/s* (the attacker-side and nameserver-side bandwidth is only *140.6Kb/s*) because it has the superior attack factors. As shown in Figure 9, all of the attacking traffic is concentrated into a bursting pulse with a window of 10-50ms, other than Simple DNS+ (400ms). Since the rate-limits for PowerDNS, MaraDNS, and Dnsmasq are 500, 7, and 150, respectively, they will promptly return `ServFail` responses to the client if the query limit is exceeded (because we send 1,000 queries).

Experiments with Different Timeouts and Query Numbers. In actual attacks, attackers can only manipulate the timeout and query number values since the other factors are determined by exploited resolvers. Here, we evaluate the effect of the timeout and query number on DNSBOMB with Unbound. As listed in Table 3, when the other factors are fixed, the larger the timeout is, the higher the *BAF* will be until all queries are distributed to each 10ms timeframe. Besides, as more queries are accumulated, the victim-side bandwidth will increase (displayed in Table 4). However,

Figure 10. DNSBOMB Experiment Network Bandwidth of Unbound during 10m using 1,000 DNS Queries.

generally, the increasing trend stops at 6k-8k because Unbound cannot concentrate more responses into the same pulse window, inducing continuous pulses like Figure 9(g).

Experiments with Multiple Resolvers. In Section 4.5, as one of our attack extensions, we propose coordinating multiple resolvers to conduct DNSBOMB. Here, we use 1-10 Unbound instances to demonstrate the result. As listed in Table 4, we could generate a 8.7Gb/s pulse with three instances that is three times that of a single instance. As the number of instances increases, the concentrating bandwidth will expand in a non-linear fashion. Our attack bandwidth, however, never exceeds 10Gb/s. We speculate the reason is that we utilize Docker to deploy 10-instances on a machine due to the lack of multiple physical machines to run this test. All these instances are constrained by our machine's limited hardware resources and the 10Gb/s network connection.

Long-term Experiments. To evaluate the attack stability, we run DNSBOMB against Unbound for 10m and sample packets using Wireshark [92] to depict the traffic Figure 10. For each attack round, we utilize a random subdomain to circumvent caching. Results show that DNSBOMB is quite stable and generates a 2.9Gb/s pulse every 10s, as predicted. Very few pulses fall short of 2.9Gb/s due to packet loss.

5.2. Public DNS Services

Public DNS Service List. According to the usage statistics from APNIC [4], we collect 46 widely-used public DNS services and their IP examples in Table 9 (Appendix A), 13 of which have IPv6 addresses, e.g., AdGuard and CloudFlare.

Experiment Setup. We use the same Ubuntu 22.04 host from the software experiment with public IPv4 and IPv6 addresses and a 1Gb/s network bandwidth to examine the attack factors and impact of these 46 public DNS services.

Theoretical Analysis. The testing results are listed in Table 9 (Appendix A). 36 public DNS services permit a rate-limit of over 1k pps, such as CloudFlare and OpenDNS. Several services' rate-limits are low for IPv4 but high for IPv6, such as AdGuard DNS, which is 100 for IPv4 but 2.7k for IPv6. Similarly, all timeouts are greater than 1.5s, with 20 over 5s (e.g., Level3 DNS) and 9 above 10s (e.g., Quad101 DNS). The average timeout is 5s. The query count of all services, except for OneDNS (18), is less than 10, with a mean of 4. Rather than 114, Baidu, and DNSPod DNS, all services support packet size greater than 1,232B, and 24 even allow responses of 4,096B-size. Additionally, only AdGuard, Ali, and CFIEC DNS return 1k responses within hundreds of milliseconds, whereas the others send them in

TABLE 5. DNSBOMB EXPERIMENT RESULTS OF 46 PUBLIC DNS SERVICES USING 1,000 DNS QUERIES.

Vendor	Practical Attack Bandwidth			
	Attacker-side	Victim-side	Nameserver-side	*BAF*
114DNS	92.2Kb/s	28.7Mb/s	234.4Kb/s[+]	319.3x
360 Secure DNS	269.5Kb/s	379.2Mb/s	269.5Kb/s	1,440.0x
AdGuard DNS	91.4Kb/s	57.4Mb/s	175.5Kb/s[+]	643.2x
AdGuard DNS[*]	393.8Kb/s	699.5Mb/s	756.2Kb/s[+]	1,819.0x
AhaDNS	92.2Kb/s	25.7Mb/s	92.2Kb/s	285.2x
Akamai Vantio DNS	429.7Kb/s	16.0Mb/s	218.0Kb/s	38.1x
Ali DNS	548.4Kb/s	162.4Mb/s	302.3Kb/s	303.2x
Alternate DNS	82.8Kb/s	3.2Mb/s	262.5Kb/s[+]	39.7x
Baidu DNS	362.5Kb/s	3.2Mb/s	135.9Kb/s	9.0x
CenturyLink DNS	182.8Kb/s	36.9Mb/s	135.9Kb/s	206.9x
CFIEC Public DNS[*]	406.2Kb/s	351.3Mb/s	394.5Kb/s	885.6x
CIRA Shield DNS	264.8Kb/s	904.9Mb/s	165.6Kb/s	3,498.8x
Cisco OpenDNS	264.8Kb/s	562.6Mb/s	529.7Kb/s[+]	2,175.1x
CleanBrowsing	353.1Kb/s	154.5Mb/s	218.0Kb/s	448.2x
CleanBrowsing[*]	935.2Kb/s	105.6Mb/s	282.0Kb/s	115.7x
CloudFlare DNS	706.2Kb/s	884.5Mb/s	441.4Kb/s	1,282.5x
CNNIC sDNS	1,005.5Kb/s	156.6Mb/s	225.8Kb/s	159.5x
Comodo Secure	984.4Kb/s	795.6Mb/s	482.8Kb/s	827.7x
ControlD DNS	296.9Kb/s	44.7Mb/s	216.4Kb/s	154.2x
CZ.NIC ODVR	248.4Kb/s	15.9Mb/s	519.5Kb/s[+]	65.4x
DNS for Family	1,331.2Kb/s	144.4Mb/s	1,331.2Kb/s	111.6x
DNS Forge	745.3Kb/s	155.5Mb/s	745.3Kb/s	213.7x
DNS.SB	745.3Kb/s	40.8Mb/s	296.9Kb/s	56.1x
DNS.WATCH	248.4Kb/s	638.6Mb/s	540.6Kb/s[+]	2,632.1x
DNSlify DNS	662.5Kb/s	35.0Mb/s	292.2Kb/s	54.1x
DNSPod Public DNS	331.2Kb/s	398.3Mb/s	274.2Kb/s	1,231.1x
Dyn DNS	362.5Kb/s	383.1Mb/s	271.9Kb/s	1,082.2x
FDN DNS	414.1Kb/s	166.9Mb/s	310.9Kb/s	412.7x
Google DNS	557.8Kb/s	32.1Mb/s	1,740.8Kb/s[+]	58.9x
G-Core DNS	828.1Kb/s	117.6Mb/s	181.2Kb/s	145.4x
HE DNS	745.3Kb/s	39.9Mb/s	218.0Kb/s	54.8x
Level3 DNS	579.7Kb/s	772.2Mb/s	283.6Kb/s	1,364.1x
LibreDNS	910.9Kb/s	31.1Mb/s	218.0Kb/s	35.0x
Neustar UltraDNS	248.4Kb/s	261.1Mb/s	689.1Kb/s[+]	1,076.1x
NextDNS	414.1Kb/s	401.1Mb/s	689.1Kb/s[+]	992.0x
Norton ConnectSafe	248.4Kb/s	256.8Mb/s	248.4Kb/s	1,058.6x
OneDNS	82.0Kb/s	24.3Mb/s	264.8Kb/s[+]	303.1x
OpenNIC DNS	165.6Kb/s	97.2Mb/s	262.5Kb/s[+]	600.9x
Quad101 DNS	82.8Kb/s	76.8Mb/s	309.4Kb/s[+]	949.5x
Quad9 DNS	331.2Kb/s	71.9Mb/s	619.5Kb/s[+]	222.3x
SafeDNS	257.8Kb/s	86.4Mb/s	590.6Kb/s[+]	343.1x
SafeSurfer DNS	662.5Kb/s	19.4Mb/s	331.2Kb/s	30.0x
SkyDNS	257.8Kb/s	118.4Mb/s	687.5Kb/s[+]	470.2x
Strongarm DNS	165.6Kb/s	4.9Mb/s	218.0Kb/s[+]	30.0x
Tiarap Public DNS	82.8Kb/s	569.1Mb/s	155.5Kb/s[+]	7,037.5x
Verisign Public DNS	248.4Kb/s	329.4Mb/s	459.4Kb/s[+]	1,357.6x
xTom DNS	662.5Kb/s	18.5Mb/s	320.3Kb/s	28.5x
Yandex DNS	82.8Kb/s	876.2Mb/s	536.7Kb/s[+]	10,834.0x

1. The query sending time and response packet size are set to the maximum values listed in Table 9. [*]: Via IPv6. Ordered by the alphabet of vendors.
2. The nameserver's receiving bandwidth is $\leq$ attacker-side's since resolvers retry < 20 times except for [+] (many backend IPs for load balancing).
3. The nameserver's sending bandwidth can be negligible since it only needs to return one response for the final query.

TABLE 6. DNSBOMB EXPERIMENT RESULTS OF 13 PUBLIC DNS SERVICES WITH DEFRAGMENTATION TIMEOUT VIA IPV6.

Vendor	Practical Attack Bandwidth			
	Attacker-side		Victim-side	*BAF*
	Original	With Frag.		
AdGuard DNS	393.8Kb/s	262.5Kb/s	491.0Mb/s	1,915.2x
CFIEC Public DNS	406.2Kb/s	293.0Kb/s	3.9Mb/s	13.7x
CleanBrowsing	935.2Kb/s	535.9Kb/s	43.4Mb/s	83.0x
CloudFlare DNS	831.2Kb/s	481.2Kb/s	630.8Mb/s	1,342.3x
ControlD DNS	296.9Kb/s	262.5Kb/s	41.5Mb/s	161.8x
DNS for Family	1,566.8Kb/s	645.3Kb/s	16.7Mb/s	26.6x
DNS Forge	877.2Kb/s	590.6Kb/s	53.3Mb/s	92.4x
DNS.SB	877.2Kb/s	535.9Kb/s	46.4Mb/s	88.7x
DNS.WATCH	292.4Kb/s	262.5Kb/s	126.2Mb/s	492.2x
FDN DNS	487.4Kb/s	317.2Kb/s	39.3Mb/s	126.8x
G-Core DNS	974.7Kb/s	481.2Kb/s	107.6Mb/s	229.0x
HE DNS	877.2Kb/s	535.9Kb/s	94.8Mb/s	181.1x
SafeDNS	303.4Kb/s	259.4Kb/s	190.3Mb/s	751.2x
Average Bandwidth	701.5Kb/s	420.2Kb/s	**40% Reduced**	

The query sending time and response packet size are set to the maximum values listed in Table 9. [*]: Via IPv6. Ordered by the alphabet of vendors.

tens of milliseconds. Based on these factors, all services are exploitable to conduct powerful DNSBOMB attacks.

Practical Attacks. We perform small-scale experiments to assess the actual attack effect of 46 public DNS service listed in Table 5. The average attacker-side bandwidth is only *426.4Kb/s*, while the victim-side bandwidth is in the *hundreds of Mb/s* (25 services). For example, CIRA Shield, CloudFlare, and Yandex DNS generate a near *900Mb/s* DoS pulse. Because our public network link is only 1Gb/s, we believe the actual pulse magnitude can be greater. 14 services have a *BAF* greater than 1k, outperforming the state-of-the-art PDoS attack [39] with the same 10s timeout. Specifically, Yandex DNS supports a *BAF* of *10,834x*. As shown in Figure 13 (Appendix A), these services concentrate responses into a well-established bursting pulse, which demonstrates the universal applicability of DNSBOMB. Several services' *BAF* is low because they restrict exploited factors. We will discuss this in Section 7.2 and present mitigation solutions.

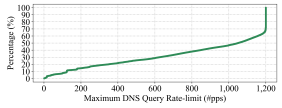
Experiments with IP Fragmentation. We introduce methods in Section 4.5 to extend the query accumulating time by leveraging IP fragmentation. Here, we present our experiment results. Before evaluation, we identify the IP defragmentation timeout and fragment number of 46 public DNS services. We find 10 services do not permit querying with fragments, such as Google and OpenDNS. Other 4 services alter Linux's default timeout to either 1s or 2s, including CloudFlare, SkyDNS, Strongarm, and Alternate DNS. Since IPv4's fragment number is only 64, we then select 13 IPv6-enabled services from the remaining 32 services for testing. Results in Table 6 show that IP fragmentation reduces the average attack-side bandwidth (*420.2Kb/s*) by 40% compared to the original (*710.5Kb/s*). We also set the IPv4 fragment number to 2,048 on a Linux machine running Unbound and do the same experiments, which reduces the bandwidth from *140.6Kb/s* to *110.9Kb/s* by 21%.

5.3. Open DNS Resolvers

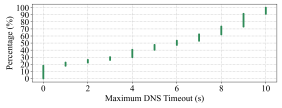
Open Resolver List. To obtain the latest resolver list [81], we scan the entire IPv4 network space for UDP port 53 using our own domain and maintain IPs returning correct answers. On July 5, 2023, we discovered over 1.8M open DNS resolvers with XMap [53] (listed in Table 7). GeoLite2

TABLE 7. OPEN RESOLVER LIST AND IDENTIFIED SOFTWARE.

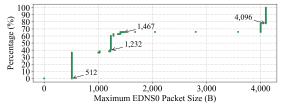
Type	Open Resolvers		
Total	Scanning on 07/05/2023	1,801,275	100.0%
Software Identified	Total	517,075	28.7%
	Microsoft	143,928	8.0%
	Dnsmasq	96,331	5.3%
	BIND	44,016	2.4%
	Unbound	15,645	0.9%
	PowerDNS	6,367	0.4%
	Simple DNS+	166	0.0%
	Knot	2	0.0%
	Others	210,619	11.7%



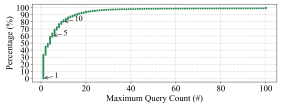
(a) Max. Rate-limit. Rate-limit Values > 1,200 are Shown as 1,200.



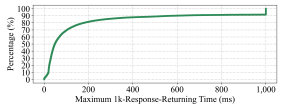
(b) Max. DNS Timeout. Timeout Values > 10s are Shown as 10s.



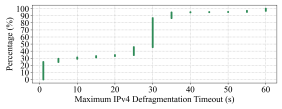
(c) Max. EDNS0 Packet Size. Size values > 4,096 are Shown as 4,096.



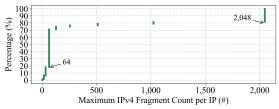
(d) Max. Query Count. Count Values > 100 are Shown as 100.



(e) Max. 1k-Responses-Returning Time. Time Values > 1s are Shown as 1s.



(f) Max. IPv4 Defragmentation Timeout. Timeout Values > 60s are Shown as 60s.



(g) Max. IPv4 Fragment Count. Count Values > 2,048 are Shown as 2,048.

Figure 11. The Distribution of Open DNS Resolver Measurement Results of DNSBOMB Attack Factors.

DB [62] indicates they belong to 227 regions and 24,795 ASes, which is sufficient proof of the vulnerable population.

Measurements. Due to ethical considerations, we only attempt to measure all attack factors of these open resolvers, instead of conducting real attack experiments like evaluating DNS software. With our own domain and nameserver, we can determine the values of attack factors. Specifically, we just identify a relative upper threshold for these factors, including 1,200 (rate-limit), 10s (timeout), 60s (IP defragmentation timeout), and 2,048 (IP fragment number), to reduce the workload. Besides, for the reason of simplicity, we regard the default EDNS0 packet size (indicated in the response's additional section like Figure 6(c)) as the maximum UDP packet size, although the actual value may be higher by setting the query's EDNS0 to a larger one.

Furthermore, to identify the software version, we employ the widely-adopted `version.bind` query (returning software version information) [6] and `fpdns` tool (a DNS software fingerprinting tool) [25]. In the end, we show practical exploiting impacts based on software testing results.

Results. The distribution of DNSBOMB attack factor measurement results is illustrated in Figure 11. Figure 11(a) shows that more than 50% of resolvers enable a rate-limit of at least 1,000, and over 80% have a rate-limit above 200. Attackers could coordinate multiple resolvers to circumvent the rate limit. Over 80% of resolvers have a >1s timeout (Figure 11(b)), which is greater than all previously exploited network path latencies (e.g., 800ms [77]). As displayed in Figure 11(c), the default EDNS0 packet size for nearly all resolvers is 512 bytes, of which 60% are larger than 1,232B and 20% are larger than 4,096B. Around 95% of resolvers transmit fewer than 20 retransmission requests to our own nameserver (Figure 11(d)), reducing the cost for attackers. As depicted in Figure 11(e), 70% can yield 1k responses to clients within 100ms. Regarding IP defragmentation factors, 10% and 30% of resolvers modify the default timeout and fragment number values, which could amplify the accumulating effect. In conclusion, all resolvers can be exploited to conduct more practical and powerful DNSBOMB attacks than previous PDoS attacks discussed in Section 2, since the accumulating time is at least the path latency.

Additionally, we identify the software version of 28.7% of 1.8M open resolvers listed in Table 7, including Microsoft DNS (143,928), Dnsmasq (96,331), and Unbound (15,645). For example, according to our tests in Section 5.1, each of the 15,645 resolvers running Unbound could independently generate the 2.9Gb/s DNSBOMB pulse traffic. By combining them, the pulse magnitude can be increased or the pulse window can be extended, inducing ongoing DoS traffic.

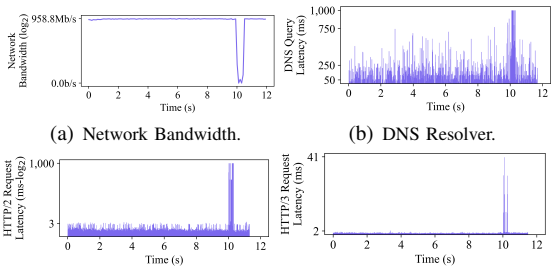
6. Real-world Evaluation of DNSBOMB Attack

In this section, we conduct controlled real-world experiments to evaluate how the DNSBOMB attack DoS target servers or degrade the RoQ of services. First, we measure the network bandwidth occupation of DNSBOMB, then use it to attack three types of servers, including a DNS resolver and HTTP/2- and HTTP/3-based websites. Results in Figure 12 show effective attack impacts that cause packet loss or significant latency for both stateless and stateful services.

Experiment Setup. We use two Unbound instances to generate combined pulsing traffic and deploy three servers with a DNS resolver and HTTP/2- and HTTP/3-based websites, respectively. All the services run on machines using Ubuntu Server 22.04 and featuring an 8-core, 2.1GHz CPU, 8GB RAM, and 1Gb/s network bandwidth. Then, we undertake the DNSBOMB attack from another machine by sending 10k queries, spoofing the IP address of these servers, and returning 4,096B-responses with a 10s timeout on the local nameserver's side. For the normal client DNS query and HTTP request, we set their client-side timeout to 1s.

6.1. Testing Network Bandwidth Occupation

To test network bandwidth occupation, we employ the `iPerf3` tool [45] that is designed for active measurements of the maximum achievable bandwidth. The minimum testing interval for it, as specified by the `-i` option, is 0.1s.



(a) Network Bandwidth. (b) DNS Resolver.

(c) HTTP/2-based Website. (d) HTTP/3-based Website.

Figure 12. DNSBOMB Attack Results on Different Servers. The Client Query or Request Timeout is Set to 1s.

As represented in Figure 12(a), the normal bandwidth decreases from 958.8Mb/s to 0.0b/s when the pulsing traffic arrives at the 10s and lasts for about 50ms, which demonstrates the actual network bandwidth DoS impact.

6.2. Attacking a DNS Resolver

We configure a DNS recursive resolver using Unbound to assess the effectiveness of DNSBOMB against stateless UDP transmissions. We send queries to UDP port 53 (under attack) every 10ms for random domains with the Golang DNS library [64] and record the query latency.

Results in Figure 12(b) indicate that the usual query latency ranges from 50 to 250ms and does not exceed 750ms. Under the DNSBOMB attack, however, the client can never receive a valid response (1s latency). This means that the query packet was lost due to DNSBOMB and the resolver could never see it. We confirmed this by inspecting captured traffic that lacked a response packet.

6.3. Attacking an HTTP/2-based Website

Apart from stateless connections, we attack stateful TCP services with DNSBOMB. We set up an Apache server [3] hosting an HTTP/2-based website. Every 10ms we use the Golang HTTP library [32] to send HTTP requests to TCP port 80 (under attack) via IP and save the request latency.

Figure 12(c) displays that the normal HTTP request time cost is only less than 3ms. Once the DNSBOMB pulsing traffic arrives, the request response from TCP port 80 (under attack) is returned at a cost of over 750ms or is lost entirely. The TCP-based HTTP service is severely degraded.

6.4. Attacking an HTTP/3-based Website

In addition, we attempt to evaluate the impact of DNSBOMB on a newly-emerged HTTP/3 standard based on QUIC [20]. HTTP/3 or QUIC is designed to decrease the effects of packet loss. We deploy an HTTP/3-based website with Caddy Server [12] that enables HTTP/3 and register a domain to set up the HTTP/3 config with Caddy's self-signed certificate [13]. Every 10ms we use the Golang HTTP library to send HTTP request to UDP port 443 (under attack) via registered domain and save the request latency.

As illustrated in Figure 12(d), when the pulsing traffic occurs, the request response increases by 2,000% from 2ms to 41ms. Compared with the TCP-based HTTP service, the QUIC-based HTTP service is partially degraded due to QUIC's improved retransmission mechanism. Specifically, QUIC's efficient per-packet retransmission mechanism, its ability to multiplex without head-of-line blocking, and its agile connection migration contribute to greater resilience against packet loss and network congestion [20].

7. Discussion and Mitigation

In this section, we first describe how we address ethical considerations, then propose detailed mitigation solutions to defend against DNSBOMB with evaluation experiments, and lastly report our responsible disclosure results.

7.1. Ethical Considerations

We adhere to Menlo Report's ethical principles [48] and best network measurement practices [70] for carrying out our evaluation experiments. First, we install all 10 analyzed DNS software on our own local machines and evaluate their amplification factors via a local network link. Second, for 46 public DNS services, we test them only several times and restrict the number of queries below their rate-limits to safely perform our tests. For example, we only send 1K queries to resolvers whose rate-limit is over 150K. Third, with regard to active measurements, such as scanning DNS resolvers and probing their attack factors, we rigorously limit the scanning rate to 5,000 pps to reduce the pressure on target networks and perform random-enumerating scans with our own domains. We just identify a relative upper threshold for these factors and never attempt to reach their limits. Besides, we configure the `PTR` record and a website to show our research objective, and no opt-out request has been received. Finally, we report our findings to all relevant vendors for disclosure purposes. Notably, Akamai reached out and wanted us to test their DNS services (see below).

7.2. Mitigation Solutions

The DNSBOMB attack is made possible by the inherent DNS resolution mechanisms for guaranteeing availability, protecting security, and enhancing reliability, instead of a classic vulnerability. Therefore, to mitigate DNSBOMB, vendors have to compromise by reducing resolution performance and standardize their implementations. According to our software analysis and best configuration practices [7], [33], we provide the following mitigation solutions. *(i)* The timeout should be reduced to a modest value like 1.8s recommended by RFC 8767 [51]. *(ii)* The rate-limit should also be restricted to a small value such as AdGuard's 100. *(iii)* The EDNS0 packet size should be standardized to 1,232B [24]. *(iv)* As shown in Figure 13, the most crucial response time can be in accordance with the timeout, e.g., Akamai and SafeSurfer DNS. For instance, if the response to

TABLE 8. DNSBOMB EXPERIMENT RESULTS OF 10 DNS SOFTWARE UNDER DIFFERENT MITIGATION SOLUTIONS.

Software	Base[1]		Timeout[2]		Rate-limit[3]		Pkt. Size[4]		Res. Time[5]		All[6]	
	BAF	*%*	*BAF*	*%*	*BAF*	*%*	*BAF*	*%*	*BAF*	*%*	*BAF*	*%*
BIND	673.9x	100.0%	122.5x	18.2%	1,347.8x	200.0%	673.9x	100.0%	13.5x	2.0%	47.2x	7.0%
Unbound	21,881.1x	100.0%	2,398.5x	11.0%	4,525.6x	20.7%	4,400.5x	20.1%	45.3x	0.2%	20.2x	0.1%
PowerDNS	419.5x	100.0%	178.9x	42.6%	1,132.1x	269.9%	237.6x	56.6%	257.8x	61.4%	20.2x	4.8%
Knot	2,246.3x	100.0%	1,225.3x	54.5%	1,347.8x	60.0%	2,246.3x	100.0%	40.4x	1.8%	13.5x	0.6%
Microsoft	1,332.4x	100.0%	280.7x	21.1%	2,649.8x	198.9%	700.8x	52.6%	44.9x	3.4%	20.2x	1.5%
Technitium	3,499.8x	100.0%	2,867.6x	81.9%	4,525.6x	129.3%	4,492.6x	128.4%	467.6x	13.4%	74.1x	2.1%
Simple DNS+	66.3x	100.0%	61.7x	93.0%	726.3x	1094.8%	97.7x	147.3%	17.5x	26.3%	20.2x	30.5%
MaraDNS	18.5x	100.0%	3.1x	16.7%	37.0x	200.0%	18.5x	100.0%	18.5x	100.0%	18.5x	100.0%
Dnsmasq	3,341.8x	100.0%	624.1x	18.7%	4,546.7x	136.1%	1,033.5x	30.9%	2,728.0x	81.6%	20.5x	0.6%
CoreDNS	3,258.4x	100.0%	524.2x	16.1%	4,389.8x	134.7%	821.8x	25.2%	158.4x	4.9%	20.5x	0.6%

[1]: Base Experiment. [2]: Timeout to 1s. [3]: Rate-limit to 100. [4]: Packet Size to 1,232. [5]: Response-Returning Time to Timeout. [6]: All Restrictions Set.

a query costs 5s, resolvers can delay 5s and then return the response to clients. Furthermore, we discover some sophisticated defensive strategies through testing. *(v)* Google limits the total amount of traffic sent to the same IP within the timeout (Figure 13(ac)), whose bandwidth does not increase as other factors are raised. *(vi)* Ali, DNS.SB, FDN, and xTom DNS will return `ServFail` responses in advance if receiving more queries while no response from nameservers.

To demonstrate the effect of our recommended mitigation solutions, we conduct 6 local experiments with different restrictions, including the base (no restrictions with 1,000 queries as before), timeout set to 1s, rate-limit set to 100, packet size set to 1,232B, response-returning time set to timeout, and all restrictions set. Results in Table 8 indicate that *the response time restriction has the best defensive effect, since it is the key to concentrating responses.* For example, the *BAF* of Unbound shrinks from 21,881.1x to merely 20.2x (0.1% remaining). In contrast, the rate-limit restriction is the worst, as some software's rate-limit, such as BIND, is initially low. Additionally, the timeout and packet size restrictions can reduce *BAF* to some degree.

7.3. Responsible Disclosure

Considering the ethical policy, we have responsibly informed affected parties about DNSBOMB and our findings, including 10 DNS software and 46 public DNS service vendors. We have so far received responses from 39 vendors and discussed mitigations with them, 24 of which confirmed DNSBOMB. They will have months to fix it. 10 CVE numbers were assigned to Technitium, Dnsmasq, and CoreDNS (see https://dnsbomb.net). We are still awaiting responses from other vendors. We summarize their responses below.

- **BIND** confirmed DNSBOMB and is coordinating with other vendors and us in a private DNS-OARC channel.
- **Unbound** planned to address DNSBOMB by limiting the timeout and number of waiting replies per IP (dropping queries if too many unanswered ones received).
- **PowerDNS** acknowledged the DNSBOMB issue and has been pondering internal mitigations in the resolver.
- **Knot** works a rate-limiting design to fix DNSBOMB.
- **Technitium, Dnsmasq, and CoreDNS** acknowledged DNSBOMB and reduced their default EDNS0 size to 1,232B. They are discussing other mitigations with us.
- **114DNS** and **360DNS** confirmed DNSBOMB, thanked our innovative research, and evaluated mitigations.
- **Akamai Vantio DNS** adopts a slow response-returning policy for identical queries and achieves good protection. We tested their services after they reached out.
- **CZ.NIC ODVR** used Knot, while **DNS.SB** and **xTom DNS** adopted PowerDNS. They are waiting for patches.
- **OneDNS** has arranged relevant mitigation measures by reducing the query rate limit and adopting a firewall.
- **Quad9 DNS** considered DNSBOMB as a very interesting attack and is discussing mitigations with us.
- **SafeDNS** was very interested in collaborating with us and wanted to schedule a further discussion.
- **AdGuard DNS**, **Ali DNS**, **Baidu DNS**, **ByteDance DNS**, **CFIEC Public DNS**, **ControlD DNS**, **Dyn DNS**, and **Yandex DNS** are now evaluating or implementing our recommended mitigations after discussing with us.

8. Conclusion

We present DNSBOMB, a novel pulsing DoS attack that exploits DNS mechanisms, transforming low-rate queries into large-sized, concentrated response bursts that can overwhelm target systems. Our small-scale evaluation shows that DNSBOMB surpasses previous PDoS attacks, with a peak pulse magnitude of 8.7Gb/s and a bandwidth amplification factor over 20,000. DNSBOMB leads to complete packet loss or service degradation in our controlled experiments.

Acknowledgement

We thank all the anonymous reviewers and our shepherd for their valuable comments and especially for our shepherd's thoughtful and patient guidance in helping us to improve this paper. Authors were supported by the National Natural Science Foundation of China (U1836213, U19B2034, 62102218, and 62132011).

References

[1] Sumayah Alrwais, Xiaojing Liao, Xianghang Mi, Peng Wang, XiaoFeng Wang, Feng Qian, Raheem Beyah, and Damon McCoy. Under the Shadow of Sunshine: Understanding and Detecting Bulletproof Hosting on Legitimate Service Provider Networks. In *S&P '17*, 2017.

[2] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *USENIX Security '17*, 2017.

[3] Apache. Apache. https://httpd.apache.org/, 2023.

[4] APNIC. DNS Resolvers Use. https://stats.labs.apnic.net/rvrs, 2023.

[5] BIND. Comparative Resolver Performance Results. https://www.isc.org/blogs/bind-resolver-performance-july-2021/, 2021.

[6] BIND. How Do I Change the Version that BIND Reports When Queried for version.bind? https://kb.isc.org/docs/aa-00359, 2021.

[7] BIND. Configuration Reference. https://bind9.readthedocs.io/en/latest/reference.html, 2022.

[8] BIND. resolver-query-timeout. https://bind9.readthedocs.io/en/latest/reference.html#namedconf-statement-resolver-query-timeout, 2023.

[9] BIND. resolver-retry-interval. https://bind9.readthedocs.io/en/latest/reference.html#namedconf-statement-resolver-retry-interval, 2023.

[10] Anat Bremler-Barr, Eli Brosh, and Mor Sides. DDoS Attack on Cloud Auto-Scaling Mechanisms. In *INFOCOM '17*, 2017.

[11] Jonas Bushart. Optimizing Recurrent Pulsing Attacks using Application-Layer Amplification of Open DNS Resolvers. In *Woot '18*, 2018.

[12] Caddy. Caddy Server. https://caddyserver.com/, 2023.

[13] Caddy. How to set up HTTP3. https://caddy.community/t/how-to-set-up-http3-on-a-localhost/18216, 2023.

[14] CAIDA. IP Spoofing. https://spoofer.caida.org/summary.php, 2023.

[15] Jiahao Cao, Qi Li, Renjie Xie, Kun Sun, Guofei Gu, Mingwei Xu, and Yuan Yang. Disrupting the SDN Control Channel via Shared Links: Attacks and Countermeasures. In *USENIX Security '19*, 2019.

[16] Wei Chen, Yingzhou Zhang, and Yuanchun Wei. The Feasibility of Launching Reduction of Quality(RoQ) Attacks in 802.11 Wireless Networks. In *ICPADS '08*, 2008.

[17] Kenjiro Cho, Kensuke Fukuda, Vivek Pai, Neil Spring, Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *IMC '15*, 2015.

[18] Cloudflare. DDoS Reports. https://blog.cloudflare.com/tag/ddos-reports/, 2023.

[19] Cloudflare. What is a Denial-of-Service (DoS) attack? https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/, 2023.

[20] CloudFlare. What is HTTP/3? https://www.cloudflare.com/learning/performance/what-is-http3/, 2023.

[21] Jakub Czyz, Michael Kallitsis, Manaf Gharaibeh, Christos Papadopoulos, Michael Bailey, and Manish Karir. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *IMC '14*, 2014.

[22] Joao Damas, Michael Graff, and Paul Vixie. RFC 6891: Extension Mechanisms for DNS (EDNS(0)). *RFC Internet Standard*, 2013.

[23] Stephen E. Deering and Robert M. Hinden. RFC 8200: Internet Protocol, Version 6 (IPv6) Specification. *RFC Internet Standard*, 2017.

[24] DNS-OARC. DNS Flag Day 2020. http://www.dnsflagday.net/2020/, 2020.

[25] DNS-OARC. Fpdns. https://www.dns-oarc.net/tools/fpdns, 2021.

[26] Wesley M. Eddy. RFC 9293: Transmission Control Protocol (TCP). *RFC Internet Standard*, 2022.

[27] Fandom. Dragon Ball Wiki. https://dragonball.fandom.com/wiki/Kamehameha, 2023.

[28] Kaiming Fang and Guanhua Yan. Paging Storm Attacks against 4G/LTE Networks from Regional Android Botnets: Rationale, Practicality, and Implications. In *WiSec '20*, 2020.

[29] Zhenqian Feng, Bing Bai, Baokang Zhao, and Jinshu Su. Shrew Attack in Cloud Data Center Networks. In *MSN '11*, 2011.

[30] Giacomo Giuliari, Tommaso Ciussani, Adrian Perrig, and Ankit Singla. ICARUS: Attacking Low Earth Orbit Satellite Networks. In *USENIX ATC '21*, 2021.

[31] GoDaddy. GoDaddy. https://www.verisign.com/, 2023.

[32] Golang. Golang HTTP. https://pkg.go.dev/net/http, 2023.

[33] Google. Google Public DNS for ISPs (Rate-limit Configuration). https://developers.google.com/speed/public-dns/docs/isp, 2023.

[34] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. Scan, Test, Execute: Adversarial Tactics in Amplification DDoS Attacks. In *CCS '21*, 2021.

[35] Liyi Gu, Jun Zhang, and Brahim Bensaou. Unleashing the Shrew: A Stealth Greedy Targeted Attack on TCP Traffic in Wireless LAN. In *LCN '14*, 2014.

[36] Mina Guirguis, Azer Bestavros, Ibrahim Matta, and Yuting Zhang. Reduction of Quality (RoQ) Attacks on Internet End-Systems. In *INFOCOM '05*, 2005.

[37] Mina Guirguis, Azer Bestavros, Ibrahim Matta, and Yuting Zhang. Reduction of Quality (RoQ) Attacks on Dynamic Load Balancers: Vulnerability Assessment and Design Tradeoffs. In *INFOCOM '07*, 2007.

[38] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *IMC '02*, 2002.

[39] Run Guo, Jianjun Chen, Yihang Wang, Keran Mu, Baojun Liu, Xiang Li, Chao Zhang, Haixin Duan, and Jianping Wu. Temporal CDN-Convex Lens: A CDN-Assisted Practical Pulsing DDoS Attack. In *USENIX Security '23*, 2023.

[40] Yanxiang He, Qiang Cao, Yi Han, Libing Wu, and Tao Liu. Reduction of Quality (RoQ) Attacks on Structured Peer-to-Peer Networks. In *IPDPS '09*, 2009.

[41] Amir Herzberg and Haya Shulman. Stealth-MITM DoS Attacks on Secure Channels. In *NDSS '09*, 2009.

[42] Amir Herzberg and Haya Shulman. Socket Overloading for Fun and Cache-Poisoning. In *ACSAC '13*, 2013.

[43] Tomas Hlavacek, Philipp Jeitner, Donika Mirdita, Haya Shulman, and Michael Waidner. Stalloris: RPKI Downgrade Attack. In *USENIX Security '22*, 2022.

[44] Allen Householder and Ian A Finlay. Various DNS Service Implementations Generate Multiple Simultaneous Queries for the Same Resource Record. https://www.kb.cert.org/vuls/id/457875, 2002.

[45] iPerf. iPerf. https://iperf.fr/, 2023.

[46] Min Suk Kang, Soo Bum Lee, and V. D. Gligor. The Crossfire Attack. In *S&P '13*, 2013.

[47] Yu-Ming Ke, Chih-Wei Chen, Hsu-Chun Hsiao, Adrian Perrig, and Vyas Sekar. CICADAS: Congesting the Internet with Coordinated and Decentralized Pulsating Attacks. In *AsiaCCS '16*, 2016.

[48] Erin Kenneally and David Dittrich. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. *SSRN Electronic Journal*, 2012.

[49] Lukas Krämer, Johannes Krupp, Daisuke Makita, Tomomi Nishizoe, Takashi Koide, Katsunari Yoshioka, and Christian Rossow. AmpPot: Monitoring and Defending Against Amplification DDoS Attacks. In *RAID '15*, 2015.

[50] Aleksandar Kuzmanovic and Edward W Knightly. Low-Rate TCP-Targeted Denial of Service Attacks: The Shrew vs. the Mice and Elephants. In *SIGCOMM '03*, 2003.

[51] David C Lawrence, Warren "Ace" Kumari, and Puneet Sood. RFC 8767: Serving Stale Data to Improve DNS Resiliency. *RFC Proposed Standard*, 2020.

[52] Xiang Li, Baojun Liu, Xuesong Bai, Mingming Zhang, Qifan Zhang, Zhou Li, Haixin Duan, and Qi Li. Ghost Domain Reloaded: Vulnerable Links in Domain Name Delegation and Revocation. In *NDSS '23*, 2023.

[53] Xiang Li, Baojun Liu, Xiaofeng Zheng, Haixin Duan, Qi Li, and Youjun Huang. Fast IPv6 Network Periphery Discovery and Security Implications. In *DSN '21*, 2021.

[54] Xiang Li, Chaoyi Lu, Baojun Liu, Qifan Zhang, Zhou Li, Haixin Duan, and Qi Li. The Maginot Line: Attacking the Boundary of DNS Caching Protection. In *USENIX Security '23*, 2023.

[55] Xiang Li, Wei Xu, Baojun Liu, Mingming Zhang, Zhou Li, Jia Zhang, Deliang Chang, Xiaofeng Zheng, Chuhan Wang, Jianjun Chen, Haixin Duan, and Qi Li. TuDoor Attack: Systematically Exploring and Exploiting Logic Vulnerabilities in DNS Response Pre-processing with Malformed Packets. In *S&P '24*, 2024.

[56] Linux. IP sysctl. https://docs.kernel.org/networking/ip-sysctl.html, 2023.

[57] Xiapu Luo and Rocky K. C. Chang. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. In *NDSS '05*, 2005.

[58] Xiapu Luo and Rocky K C. Chang. Optimizing the Pulsing Denial-of-Service Attacks. In *DSN '05*, 2005.

[59] Gabriel Maciá-Fernández, Jesús E. Díaz-Verdejo, and Pedro García-Teodoro. Assessment of a Vulnerability in Iterative Servers Enabling Low-Rate DoS Attacks. In *ESORICS '06*, 2006.

[60] Gabriel Maciá-Fernández, Jesús E. Díaz-Verdejo, and Pedro García-Teodoro. Mathematical Model for Low-Rate DoS Attacks Against Application Servers. *TIFS '09*, 2009.

[61] Gabriel Maciá-Fernández, Jesús E. Díaz-Verdejo, Pedro García-Teodoro, and Francisco de Toro-Negro. LoRDAS: A Low-Rate DoS Attack against Application Servers. In *CRITIS '07*, 2007.

[62] MaxMind. GeoLite2 Free Geolocation Data. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data, 2023.

[63] Xianghang Mi, Xuan Feng, Xiaojing Liao, Baojun Liu, XiaoFeng Wang, Feng Qian, Zhou Li, Sumayah Alrwais, Limin Sun, and Ying Liu. Resident Evil: Understanding Residential IP Proxy as a Dark Service. In *S&P '19*, 2019.

[64] Miekg. Go DNS. https://github.com/miekg/dns, 2023.

[65] Paul V. Mockapetris. RFC 1034: Domain Names - Concepts and Facilities. *RFC Internet Standard*, 1987.

[66] Paul V. Mockapetris. RFC 1035: Domain Names - Implementation and Specification. *RFC Internet Standard*, 1987.

[67] Namecheap. Namecheap Domain Registration. https://www.namecheap.com/, 2023.

[68] NIST. CVE-2002-2211. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-2211, 2002.

[69] Jeman Park, DaeHun Nyang, and Aziz Mohaisen. Timing is Almost Everything: Realistic Evaluation of the Very Short Intermittent DDoS Attacks. In *PST '18*, 2018.

[70] Craig Partridge and Mark Allman. Ethical Considerations in Network Measurement Papers. *Communications of the ACM*, 2016.

[71] Vern Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *SIGCOMM CCR '01*, 2001.

[72] Vern Paxson, Mark Allman, H.K. Jerry Chu, and Matt Sargent. RFC 6298: Computing TCP's Retransmission Timer. *RFC Proposed Standard*, 2011.

[73] Jon Postel. RFC 768: User Datagram Protocol. *RFC Internet Standard*, 1980.

[74] Jon Postel. RFC 791: Internet Protocol. *RFC Internet Standard*, 1981.

[75] Jon Postel. RFC 793: Transmission Control Protocol. *RFC Internet Standard*, 1981.

[76] PowerDNS. High Performance, Low Latency, Full Flexibility. https://www.powerdns.com/powerdns-recursor, 2023.

[77] Ryan Rasti, Mukul Murthy, Nicholas Weaver, and Vern Paxson. Temporal Lensing and Its Application in Pulsing Denial-of-Service Attacks. In *S&P '15*, 2015.

[78] Wei Ren, Hai Jin, and Tenghong Liu. Congestion Targeted Reduction of Quality of Service DDoS Attacking and Defense Scheme in Mobile Ad Hoc Networks. In *ISM'05*, 2005.

[79] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In *IMC '14*, 2014.

[80] RIPE. RIPE Atlas. https://atlas.ripe.net/, 2023.

[81] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. On measuring the client-side DNS infrastructure. In *IMC '13*, 2013.

[82] Suhas Setikere, Vinay Sachidananda, and Yuval Elovici. Out of Kilter: Holistic Exploitation of Denial of Service in Internet of Things. In *SecureComm '18*, 2018.

[83] Huasong Shan, Qingyang Wang, and Calton Pu. Tail Attacks on Web Applications. In *CCS '17*, 2017.

[84] Huasong Shan, Qingyang Wang, and Qiben Yan. Very Short Intermittent DDoS Attacks in an Unsaturated System. In *SecureComm '17*, 2017.

[85] A. Shevtekar, J. Stille, and N. Ansari. On the Impacts of low rate DoS attacks on VoIP traffic. In *SCN '08*, 2008.

[86] Sooel Son and Vitaly Shmatikov. The Hitchhiker's Guide to DNS Cache Poisoning. In *SecureComm '10*, 2010.

[87] Ahren Studer and Adrian Perrig. The Coremelt Attack. In *ESORICS '09*, 2009.

[88] Yuta Takahashi, Hiroshi Inamura, and Yoshitaka Nakamura. A Low-rate DDoS Strategy for Unknown Bottleneck Link Characteristics. In *PerCom '21*, 2021.

[89] Yajuan Tang, Xiapu Luo, Qing Hui, and Rocky K. C. Chang. Modeling the Vulnerability of Feedback-Control Based Internet Services to Low-Rate DoS Attacks. *TIFS '13*, 2014.

[90] Johanna Ullrich and Edgar Weippl. The Beauty or The Beast? Attacking Rate Limits of the Xen Hypervisor. In *ESORICS '16*, 2016.

[91] Unbound. max-query-restarts. https://unbound.docs.nlnetlabs.nl/en/latest/manpages/unbound.conf.html#unbound-conf-max-query-restarts, 2023.

[92] Wireshark. Wireshark. https://www.wireshark.org/, 2023.

[93] Wei Xu, Xiang Li, Chaoyi Lu, Baojun Liu, Jia Zhang, Jianjun Chen, Tao Wan, and Haixin Duan. TsuKing: Coordinating DNS Resolvers and Queries into Potent DoS Amplifiers. In *CCS '23*, 2023.

[94] Meng Yue, Minxiao Wang, and Zhijun Wu. Low-High Burst: A Double Potency Varying-RTT Based Full-Buffer Shrew Attack Model. *TDSC '21*, 2021.

[95] Qifan Zhang, Xuesong Bai, Xiang Li, Haixin Duan, Qi Li, and Zhou Li. ResolverFuzz: Automated Discovery of DNS Resolver Vulnerabilities with Query-Response Fuzzing. In *USENIX Security '24*, 2024.

[96] Ying Zhang, Zhuoqing Mao, and Jia Wang. Low-Rate TCP-Targeted DoS Attack Disrupts Internet Routing. In *NDSS '07*, 2007.

Appendix A. Experiments of Public DNS Services.

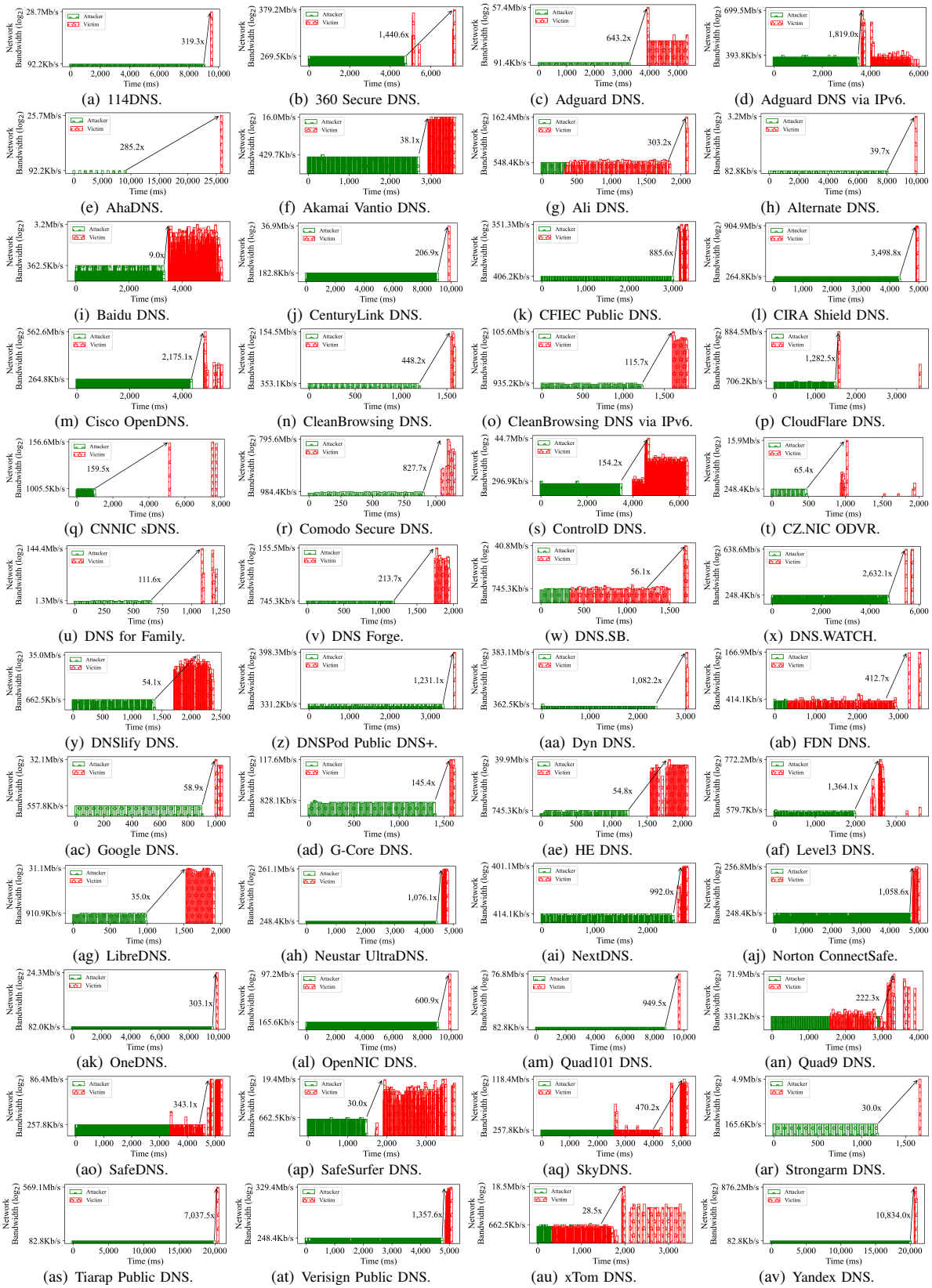
Table 9 lists 46 widely-used, evaluated public DNS services, as well as identified DNSBOMB attack factor results and theoretical analysis.

Figure 13 shows the practical attack bandwidth results of 46 public DNS services.

TABLE 9. DNS RESOLUTION MECHANISMS RELATED TO DNSBOMB AND THEORETICAL ANALYSIS OF 46 PUBLIC DNS SERVICES.

Public DNS Service		Accumulating		Amplifying				Concentrating	Theoretical Analysis			
Vendor	IP Example	Rate-limit (#)	Time-out (s)	Query (#)*	EDNS0 (B)	Max. Size (B) O. E.[1]	Max. Size (B) W. E.[2]	Returning Time (ms)[3]	*PCE*[4]	*SCE*[5]	*TCE*[6]	*BCE*
114DNS	114.114.114.114	198	10	5	512	512	523	5	1x	5.2x	1,000.0x	5,230.0x
360 Secure DNS	101.226.4.6	150k+	6	4	2,048	512	2,048	30	1x	20.5x	200.0x	4,096.0x
AdGuard DNS	94.140.14.14	100	4	4	0	512	1,463	100	1x	14.6x	40.0x	585.2x
AdGuard DNS	2a10:50c0::ad1:ff	2.7k	4	4	0	512	4,096	100	1x	41.0x	40.0x	1,638.4x
Akamai Vantio DNS	23.56.160.142	10k+	3	4	512	512	4,084	50	1x	40.8x	60.0x	2,450.4x
AhaDNS	5.2.75.75	11	30	1	1,472	512	4,096	1	1x	41.0x	1,100.0x	45,056.0x
Ali DNS	223.5.5.5	50k+	2.5	4	1,408	512	1,396	100	1x	14.0x	25.0x	349.0x
Alternate DNS	76.76.19.19	35	10	8	4,096	512	4,096	50	1x	41.0x	200.0x	8,192.0x
Baidu DNS	180.76.76.76	50k+	4	3	-	495	495	20	1x	5.0x	200.0x	990.0x
CenturyLink DNS	205.171.3.66	2.5k	10	6	1,232	512	1,232	20	1x	12.3x	500.0x	6,160.0x
CFIEC Public DNS	240C::6644	20k+	3.5	6	1,408	489	4,084	100	1x	40.8x	35.0x	1,429.4x
CIRA Shield DNS	149.112.121.10	50k+	5	4	512	512	4,096	37	1x	41.0x	135.1x	5,535.1x
Cisco OpenDNS	208.67.222.222	25k	6	8	4,096	512	1,421	40	1x	14.2x	150.0x	2,131.5x
CleanBrowsing	185.228.168.10	400	1.5	1	512	512	1,232	30	1x	12.3x	50.0x	616.0x
CleanBrowsing	2a0d:2a00:1::	15k+	1.5	1	512	512	1,232	60	1x	12.3x	25.0x	308.0x
CloudFlare DNS	1.1.1.1	50k+	2	2	1,232	512	1,452	15	1x	14.5x	133.3x	1,936.0x
CloudFlare DNS	2606:4700:4700::1111	50k+	2	2	1,232	512	1,232	15	1x	12.3x	133.3x	1,642.7x
CNNIC sDNS	1.2.4.8	10k+	1.5	10	4,096	512	1,467	45	1x	14.7x	33.3x	489.0x
Comodo Secure	8.26.56.10	20k+	1	1	4,096	512	4,096	50	1x	41.0x	20.0x	819.2x
ControlD DNS	76.76.2.5	15k+	4.5	2	512	1,232	1,232	20	1x	12.3x	225.0x	2,772.0x
ControlD DNS	2606:1a40::5	15k+	4.5	2	512	1,232	1,232	20	1x	12.3x	225.0x	2,772.0x
CZ.NIC ODVR	193.17.47.1	120	1.5	2	1,232	512	1,232	50	1x	12.3x	30.0x	369.6x
DNS for Family	94.130.180.225	5k+	1	1	512	512	4,096	50	1x	41.0x	20.0x	819.2x
DNS for Family	2a01:4f8:1c0c:40db::1	5k+	1	1	512	512	4,096	50	1x	41.0x	20.0x	819.2x
DNS Forge	176.9.93.198	1k	1.7	1	4,096	501	1,243	50	1x	12.4x	34.0x	422.6x
DNS Forge	2a01:4f8:151:34aa::198	1k	1.7	1	4,096	501	1,243	50	1x	12.4x	34.0x	422.6x
DNS.SB	185.222.222.222	50k+	1.5	1	512	512	1,232	35	1x	12.3x	42.9x	528.0x
DNS.SB	2a09::	50k+	1.5	1	512	512	1,232	35	1x	12.3x	42.9x	528.0x
DNS.WATCH	84.200.69.80	5k+	5	8	4,096	512	4,096	60	1x	41.0x	83.3x	3,413.3x
DNS.WATCH	2001:1608:10:25::1c04:b12f	5k+	5	8	4,096	512	4,096	60	1x	41.0x	83.3x	3,413.3x
DNSlify DNS	185.235.81.1	15k+	1.5	2	512	512	1,232	50	1x	12.3x	30.0x	369.6x
DNSPod Public DNS+	119.28.28.28	50k+	4	10	4,096	485	496	20	1x	5.0x	200.0x	992.0x
Dyn DNS	216.146.35.35	600	3	8	1,372	501	1,467	20	1x	14.7x	150.0x	2,200.5x
FDN DNS	80.67.169.12	10k+	3	1	1,232	512	4,096	25	1x	41.0x	120.0x	4,915.2x
FDN DNS	2001:910:800::40	10k+	3	1	1,232	512	4,096	25	1x	41.0x	120.0x	4,915.2x
Google DNS	8.8.8.8	500	1	1	512	512	4,096	50	1x	41.0x	20.0x	819.2x
G-Core DNS	95.85.95.85	1k+	1.5	2	512	512	1,232	50	1x	12.3x	30.0x	369.6x
G-Core DNS	2a03:90c0:999d::1	1k+	1.5	2	512	512	1,232	50	1x	12.3x	30.0x	369.6x
HE DNS	74.82.42.42	1k	1.5	1	512	512	1,232	40	1x	12.3x	37.5x	462.0x
HE DNS	2001:470:20::2	1.5k	1.5	1	512	512	1,232	40	1x	12.3x	37.5x	462.0x
Level3 DNS	4.2.2.1	3.5k	6	8	4,096	512	4,096	50	1x	41.0x	120.0x	4,915.2x
LibreDNS	88.198.92.222	10k+	1.5	1	512	512	1,232	20	1x	12.3x	75.0x	924.0x
Neustar UltraDNS	156.154.70.1	20k+	5	7	4,096	512	4,096	50	1x	41.0x	100.0x	4,096.0x
NextDNS	45.90.30.118	2.5k	3	4	1,232	512	5,912	20	1x	59.1x	150.0x	8,868.0x
Norton ConnectSafe	199.85.126.10	20k+	5	1	4,096	512	4,096	20	1x	41.0x	250.0x	10,240.0x
OneDNS	52.80.66.66	540	10	18	1,232	512	1,232	50	1x	12.3x	200.0x	2,464.0x
OpenNIC DNS	103.1.206.179	15k+	10	9	1,232	512	1,232	50	1x	12.3x	200.0x	2,464.0x
Quad101 DNS	101.101.101.101	220	10	9	1,232	512	1,232	20	1x	12.3x	500.0x	6,160.0x
Quad9 DNS	9.9.9.9	5k+	3	2	1,232	512	1,232	20	1x	12.3x	150.0x	1,848.0x
SafeDNS	195.46.39.39	15k+	5	7	1,232	512	4,096	50	1x	41.0x	100.0x	4,096.0x
SafeDNS	2001:67c:2778::3939	15k+	5	7	1,232	512	4,096	50	1x	41.0x	100.0x	4,096.0x
SafeSurfer DNS	104.197.28.121	3k+	2	2	512	512	1,232	20	1x	12.3x	100.0x	1,232.0x
SkyDNS	193.58.251.251	15k+	5	2	1,232	312	4,084	50	1x	40.8x	100.0x	4,084.0x
Strongarm DNS	52.3.100.184	150	1.5	1	512	512	1,232	30	1x	12.3x	50.0x	616.0x
Tiarap Public DNS	174.138.21.128	500	20	1	1,232	512	4,096	50	1x	41.0x	400.0x	16,384.0x
Verisign Public DNS	64.6.65.6	10k+	5	7	4,096	512	4,096	20	1x	41.0x	250.0x	10,240.0x
xTom DNS	185.184.222.222	10k+	1.5	1	512	512	1,232	20	1x	12.3x	75.0x	924.0x
Yandex DNS	77.88.8.1	2.5k	20	8	4,096	512	4,096	50	1x	41.0x	400.0x	16,384.0x

[1]: Max. DNS packet size enabled for clients without EDNS0. [2]: Max. DNS packet size enabled for clients with EDNS0=4,096.
[3]: Response returning time for 1,000 packets. [4]: Number of reply/Number of query. [5]: Maximum DNS packet size/100.
[6]: Timeout/Response-returning-time (Time less than 10ms is considered as 10ms). Ordered by the alphabet of vendors.
*: The number of queries sent to the nameserver if not receiving responses during the timeout window.



(a) 114DNS. (b) 360 Secure DNS. (c) Adguard DNS. (d) Adguard DNS via IPv6.

(e) AhaDNS. (f) Akamai Vantio DNS. (g) Ali DNS. (h) Alternate DNS.

(i) Baidu DNS. (j) CenturyLink DNS. (k) CFIEC Public DNS. (l) CIRA Shield DNS.

(m) Cisco OpenDNS. (n) CleanBrowsing DNS. (o) CleanBrowsing DNS via IPv6. (p) CloudFlare DNS.

(q) CNNIC sDNS. (r) Comodo Secure DNS. (s) ControlD DNS. (t) CZ.NIC ODVR.

(u) DNS for Family. (v) DNS Forge. (w) DNS.SB. (x) DNS.WATCH.

(y) DNSlify DNS. (z) DNSPod Public DNS+. (aa) Dyn DNS. (ab) FDN DNS.

(ac) Google DNS. (ad) G-Core DNS. (ae) HE DNS. (af) Level3 DNS.

(ag) LibreDNS. (ah) Neustar UltraDNS. (ai) NextDNS. (aj) Norton ConnectSafe.

(ak) OneDNS. (al) OpenNIC DNS. (am) Quad101 DNS. (an) Quad9 DNS.

(ao) SafeDNS. (ap) SafeSurfer DNS. (aq) SkyDNS. (ar) Strongarm DNS.

(as) Tiarap Public DNS. (at) Verisign Public DNS. (au) xTom DNS. (av) Yandex DNS.

Figure 13. DNSBomb Experiment Network Bandwidth of 46 DNS Public Services using 1,000 DNS Queries.

Appendix B. Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

This paper explores a novel attack, termed DNSBomb, which leverages the security and reliability features of DNS, including query-response timeouts, query aggregation, and response fast-returning. By combining these features, an attacker can orchestrate a pulsing Denial of Service (PDoS) attack. The attack strategy involves IP-spoofing multiple DNS queries to a domain controlled by the attacker, then withholding responses to aggregate multiple replies. DNSBomb aims to overwhelm victims with periodic bursts of amplified traffic that are challenging to detect. The DNSBomb attack is demonstrated through experiments conducted on 10 DNS software, 46 public DNS services, and 1.8 million open DNS resolvers.

B.2. Scientific Contributions

- Identifies an Impactful Vulnerability
- Provides a Valuable Step Forward in an Established Field

B.3. Reasons for Acceptance

1) This paper introduces a novel vulnerability and attack approach that could open the door to a fresh line of PDoS attack work. DNSBomb leverages established, widely-adopted DNS security, reliability, and availability mechanisms to execute pulsating DoS attacks capable of substantially amplifying victim traffic.
2) The paper contains the authors have conducted a comprehensive evaluation of the attack. The evaluation is well-designed and covers various aspects, including different DNS software, public DNS services, and open DNS resolvers. It also explores theoretical versus practical attack scenarios and controlled versus real-world settings. The results of the evaluation effectively validate the practicality and potency of the attack.
3) The paper provides several mitigation strategies, and evaluates numerically the potential of those mitigations to resolve the issue.

B.4. Noteworthy Concerns

1) The reviewers note that the difference between theoretical and practical amplification results in Section 5.1 is significant. The authors have provided a satisfactory discussion behind the reason for these differences, but not all cases have been considered in detail individually.
2) The reviewers note that the effectiveness of the attack extensions may be dependent on how well multiple resolvers can be coordinated.