Stopping Amplified DNS DDoS Attacks Through Distributed Query Rate Sharing

Saurabh Verma*, Ali Hamieh†, Jun Ho Huh‡, Henrik Holm§
Siva Raj Rajagopalan‡, Maciej Korczynski**, Nina Fefferman†,
*University of Minnesota Twin Cities, verma@cs.umn.edu
†Rutgers University, {ali.adhamieh@, feffermn@dimacs.}rutgers.edu
‡Honeywell ACS Labs, {junho.huh, Siva.Rajagopalan}@honeywell.com
§Forest Glen Research, LLC, henrik@forestglenresearch.com
**Delft University of Technology, maciej.korczynski@tudelft.nl

***Abstract*—An Amplified DNS DDoS (ADD) attack involves tens of thousands of DNS resolvers that send huge volumes of amplified DNS responses to a single victim host, quickly flooding the victim's network bandwidth. Because ADD attacks are distributed, it is difficult for individual DNS resolvers to detect them based on local DNS query rates alone. Even if a victim detects an ADD attack, it cannot stop the attacker from flooding its network bandwidth. To address this problem, we present a novel mitigation system called "Distributed Rate Sharing based Amplified DNS-DDoS Attack Mitigation" (DRS-ADAM). DRS-ADAM facilitates DNS query rate sharing between DNS resolvers that are involved in an attack to detect and *completely stop* an ADD attack. Each DNS resolver quickly builds the global DNS query rate for potential victims by accumulating the shared rate values, and uses that global rate to make mitigation decisions locally. DRS-ADAM can be easily deployed through a small software update on resolvers and victim hosts, and does not require any additional server component. Our simulation results show that DRS-ADAM can contain the peak attack rates close to a victim's acceptable threshold values (which are far smaller than their sustainable bandwidth) at all times, regardless of the number of resolvers involved in ADD attacks. ADD attacks can be fully mitigated within a few seconds.**

***Index Terms*—DDoS attack, DNS amplification attack, distributed detection system.**

I. Introduction

DNS amplification-based distributed denial of service (DDoS) attacks aim to overwhelm a victim's network bandwidth to make its system resources unavailable. Such attacks amplify huge volumes of DNS responses through tens of thousands of open recursive DNS resolvers. DNS response amplification is achieved by sending a query that typically returns a large TXT record or zone file [1]. For example, an "ANY" query corresponding to 64 bytes of request can return a 3,223 bytes response, resulting in about 50x amplification. Attackers ensure that amplified responses (UDP packets) reach the victim by spoofing the victim's source IP address. We refer to such attacks as "amplified DNS DDoS" (ADD) attacks.

Successful ADD attacks can severely interrupt normal host machine operations (e.g., a web server) by depleting its bandwidth, CPU, or memory, or forcing it to shut down. Because millions of open recursive DNS resolvers around the world are available and highly accessible and the amplification can be performed easily [2], ADD attacks are now a major, popular cyber threat. Because they are distributed, ADD attacks are difficult to detect and stop. Spoofed DNS queries are sent to numerous DNS resolvers, which, from each resolver's perspective, makes the incoming DNS query rate very small. It is difficult for a resolver to detect an ADD attack based solely on its own DNS query rates.

The response rate limit (RRL) feature available on a popular DNS software called BIND [3] scrutinizes the local query rates to limit the amount of response traffic going to a potential victim host. ADD attacks can still successfully bypass RRL since it relies only on local query rates to detect the attack and uses rate-limit to mitigate it; we describe this situation in Section IV-C4. As proposed by [4], a firewall application managed by a victim host could potentially detect an ADD attack based on the incoming traffic, but the victim and its firewall cannot *stop* the attack traffic from flooding its network bandwidth.

In this paper, we propose a novel mitigation system for ADD attacks called "Distributed Rate Sharing based Amplified DNS-DDoS Attack Mitigation" (DRS-ADAM), which allows each DNS resolver to detect and stop ADD traffic locally. DNS resolvers involved in a potential ADD attack share information about incoming DNS query rates with each other, allowing each resolver to accumulate the rates and gain greater insight into the global state of the DNS responses sent out to a victim.

Our simulation results show that DRS-ADAM can contain attacks close to the threshold values at all times regardless of the number of resolvers involved in attacks or the total attack sizes. Even in partial deployment scenarios (e.g., 50% of DNS resolvers involved in an ADD attack are equipped with DRS-ADAM), DRS-ADAM can perform close to the theoretically best-possible DNS resolver-oriented (local or distributed) detection. We also used DRS-ADAM to simulate possible manipulating scenarios, including a scenario in which an attacker impersonates a victim to stop DNS resolvers from sending legitimate DNS responses to that victim. We showed that DRS-ADAM can sufficiently mitigate such attacks.

II. Motivations

ADD attack duration typically varies from 1 to 16 hours, and DDoSed financial services firms lose an average of $40,000

978-1-5090-0990-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ARES.2016.93

per hour [5]. ADD attacks increased over 132.43 percent between the second quarter of 2014 and the second quarter of 2015 [6]. Financial losses related to ADD attacks are often discussed with respect to how the victim hosts are affected, and do not discuss the losses that affect network operators that manage DNS resolvers. In this section, we explain the high level economical, monetary incentives for network operators, e.g., Internet service providers (ISPs), to deploy DRS-ADAM on their DNS resolvers, and become part of the global ADD attack detection operations.

A recent report [7] indicates that millions of home routers could be exposing ISPs and their users to become part of an ADD attack. Since the bandwidth is not free, network operators that get exposed to any kind of DDoS attack have to also pay for the bandwidth that are being abused by a DDoS attack. Rapidly growing volume and frequency of different types of DDoS attacks are significantly impacting the costs network operators have to pay for the bandwidth that is being consumed.

A quick analysis can demonstrate such costs for network operators. Hypothetically, let us assume that it costs about $\$0.50$ per Gb for an ISP (these rough figures are interpolated from [8]) to provide the promised bandwidth to their customers. Then, for an average ADD attack that abuses about 10 Gbps [9] and lasts about 60 minutes, it will roughly incur $\$18,000$ extra dollars for an ADD attack. If we assume, there are more than 100 ADD attacks being performed globally every month [10], it could potentially increase the overall network cost by $\$1,800,000$ per month; such cost would be shared by the network operators. With the deployment of DRS-ADAM, however, network operators can stop ADD attacks at their early stages (before reaching the peak attack rate), and save such costs. We will revisit this cost-saving motivation for network operators in section V-B, and demonstrate the economic benefits for network operators in terms of early attack detection, cost avoidance, and deployment costs.

ADD attacks exploit hundreds to thousands of DNS resolvers to maintain a relatively low query rate at each DNS resolver. This low rate makes it difficult for DNS resolvers to distinguish between legitimate traffic and ADD attack traffic, as their characteristics will be similar at the local level. Many existing solutions [4], [11], [12], [13] propose to detect attacks at the victim host level and then drop packets. When we consider large real-world ADD attacks that can peak around 300 Gbps [14], dropping packets at the victim host level is not a viable solution because it will not prevent capacity from being consumed.

In contrast, DRS-ADAM detects and stops the attack at its origin, i.e., at the DNS resolver level. DNS resolvers involved in an attack share DNS query rate information with each other, building an accumulated query rate and making mitigation decisions locally. DRS-ADAM design was motivated by questions like "Is it possible to detect and stop ADD attacks in a timely manner by facilitating DNS query sharing between DNS resolvers?," "Can ADD attacks be mitigated or stopped before they reach victim's sustainable bandwidth?," and "How much network overhead would DNS query rate sharing incur on resolvers?" In Section V, we answer these questions with respect to the simulation results.

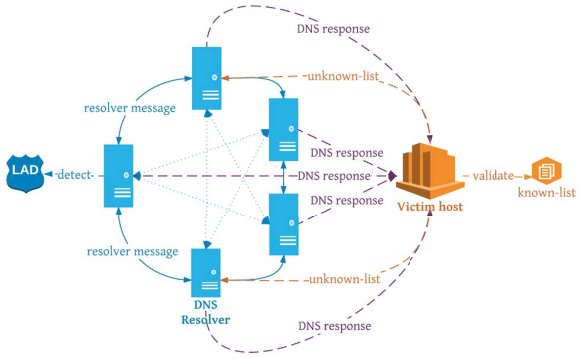


Fig. 1: Architecture of DRS-ADAM.

One might consider a solution where the victim directly requests individual DNS resolvers to stop sending response packets once it witnesses a few mismatching DNS queries. But without a robust authentication mechanism in place (e.g. PKI)—which can be expensive—resolvers would not be able to verify the identify of the victim. This kind of solution could potentially be abused by adversaries to perform another type of DoS attack—sending fake "stop" requests to resolvers, and stopping legitimate DNS responses from reaching the victim.

III. DRS-ADAM DESIGN

We describe DRS-ADAM and the protocols used between DNS resolvers and between DNS resolvers and victim.

A. Design goals

Four design goals informed DRS-ADAM development. (1) Our first design goal was to develop a solution that is capable of containing ADD attacks under victim-specified, acceptable bandwidth threshold values at all times, and completely stopping ADD attacks eventually regardless of the number of resolvers involved in an attack. (2) Our next design goal was to develop a software-based solution that uses existing DNS resolvers to achieve high deployability and scalability. We wanted to avoid introducing new, dedicated intrusion detection system servers or other hardware infrastructure. Roughly 27 million DNS resolvers are connected to the Internet worldwide; introducing new, dedicated IDS servers would incur unmanageable costs and be difficult to scale. (3) Our third design goal was to develop a solution that works purely on UDP. Most DNS services are provided and consumed over UDP, which consumes minimum memory/overhead and does not respond to congestion. These characteristics allow fast ADD detection. (4) Our last goal was to avoid expensive authentication techniques that use public-key infrastructure (PKI) for authenticating DNS resolvers or victim hosts.

B. Architecture overview

The three main components that make DRS-ADAM work are DNS resolvers, local anomaly detectors (LADs) running

inside DNS resolvers, and a UDP-based DNS query rate sharing protocol. Figure 1 represents the DRS-ADAM architecture depicting UDP-based communications between DNS resolvers and between a victim's firewall and DNS resolvers.

A firewall protecting a victim maintains a predefined list of known DNS resolver IP addresses called the "known-list" that keeps track of the DNS resolvers with which the victim sent a DNS query. A small piece of code running on the firewall monitors incoming DNS response messages and checks the source IP address against the addresses stored in the known-list. Note, if a firewall is not present, the victim itself can install the code and monitor incoming DNS responses. If an unknown source IP address is detected, the firewall adds that address to the "unknown-list," which keeps track of unknown DNS resolvers that are potentially involved in an ADD attack. Section III-C describes in detail the protocol used by the victim's firewall and resolvers to optimally share the unknown-list. The query rate sharing protocol the resolvers use is described in Section III-D. The local anomaly detectors running inside the resolvers continuously compute accumulated shared query rates to make mitigation decisions locally.

In the initial presentation of the DRS-ADAM we made two simplifying assumptions for clarity. We assumed that DNS resolvers and victim hosts and firewalls are all discoverable and can communicate with each other. Our second assumption was that, since DRS-ADAM relies on collecting victim's DNS query from the resolvers, we expect that collected DNS query rates do not change much during the attack period. Both assumptions can be relaxed by moving from the complete, global network version of resolver communication to compute accumulated query rates to a version that uses an optimized distributed sum computation (described in Section V-F).

C. Communication between the victim and DNS resolvers

When an unknown DNS response (that the victim has not requested) arrives, the victim adds the IP address of the DNS resolver that sent the response in its `unknown-list`— if the IP address does not already exist. Every time a new IP address is added, the victim sends only the new IP address to the second-to-last resolver listed in the `unknown-list`.

TABLE I: Notations used in the communication protocols

V	: A victim's upgraded firewall.
D_i	: i^{th} DNS resolver IP address entry in `unknown-list`.
DIP_i	: The IP address of the i^{th} DNS resolver.
`unknown-list`	: A list of unknown DNS resolver IP addresses (those that are possibly involved in attack).
SDR_i	: A spoofed DNS response sent to victim from i^{th} DNS resolver.
$threshold$	: Threshold absolute value sent by the victim.

Based on the notations from Table I, let us assume at $t = 0$ a victim host V receives a DNS response from D_i, which is not listed in either the `known-list` or `unknown-list`. V creates a new entry and appends the IP address of D_i to the `unknown-list`. V then forwards D_i's IP address to only the DNS resolver listed as the second to last resolver in the `unknown-list`. In Section III-D, we explain why that process is sufficient for DNS resolvers to build the complete network of resolvers involved in the attack.

If D_i is already known, V does not take any action. That protocol is described below.

$D_i \longrightarrow V$	$: SDR_i$	Victim V receives a spoofed DNS response from D_i for the first time.
$V \longrightarrow D_{i-1}$	$: DIP_i, threshold$	Victim V forwards threshold and IP address of D_i to D_{i-1}.

V also shares its "absolute threshold value" with D_{i-1}, which is determined based on the victim's experience of the expected DNS traffic and sustainable bandwidth. Note that DNS traffic volume for enterprises is typically less than 1% of the total traffic [15]. The protocol was designed to minimize communication complexity and the bandwidth consumed by the victim. A victim only needs to send a single IP address each time its `unknown-list` gets updated. With n number of resolvers involved in an ADD attack, the total complexity of messages forwarded to resolvers will be $O(n)$.

D. Query rate sharing between DNS resolvers

We designed a protocol that allows resolvers to quickly discover all other resolvers involved in an ADD attack while consuming an acceptable amount of bandwidth. That protocol is described below.

$D_{n+1} \longrightarrow V$	$: SDR_n$	V receives a spoofed DNS response from D_{n+1} for the first time.
$V \longrightarrow D_n$	$: DIP_{n+1}$	V forwards D_{n+1}'s IP address to D_n.
$D_n \longrightarrow D_1$	$: DIP_{n+1}$	D_n forwards D_{n+1}'s IP address to D_1.
$D_n \longrightarrow D_2$	$: DIP_{n+1}$	D_n forwards D_{n+1}'s IP address to D_2.
$\vdots \longrightarrow \vdots$	$: \vdots$	
$D_n \longrightarrow D_{n-1}$	$: DIP_{n+1}$	D_n forwards D_{n+1}'s IP address to D_{n-1} resolver.
$D_n \longrightarrow D_{n+1}$	$: DnsList$	D_n forwards its current full `unknown-list` to D_{n+1}. resolver.

Assume that at time t a victim has an `unknown-list` L_n containing n number of DNS resolvers, namely $\{D_1, D_2, \ldots, D_n\}$. At the same time, all of the listed DNS resolvers $\{D_1, D_2, \ldots, D_n\}$ maintain the same L_n. At time $t + 1$ a new DNS resolver D_{n+1} gets added to L_n, and the victim updates the list to L_{n+1}. The victim sends the IP address of the new D_{n+1} just to the second last DNS resolver in L_{n+1}, which is D_n. Upon receiving that IP address, D_n sends the IP address of D_{n+1} to all of $\{D_1, D_2, \ldots, D_n\}$, and also sends the full `unknown-list` to the newly added D_{n+1}. After that, all DNS resolvers $\{D_1, \ldots, D_{n+1}\}$ have L_{n+1} = $\{D_1, D_2, \ldots, D_{n+1}\}$. This process follows iteratively for the remaining resolvers involved in the attack. We refer to messages that carry information related to the `unknown-list` as "resolver messages." These messages are UDP packets, as planned in our design goals (see Section III-A).

The D_i resolver may receive a resolver message that contains the IP address of D_{i+1} from the victim before it receives the full `unknown-list` from D_{i-1}. This case can be handled by making the D_i resolver solely responsible for propagating the D_{i+1} resolver IP to everyone in the `unknown-list` as it is continuously updated. This is achieved by simply keeping track of the `unknown-list` resolver IPs to which D_i has sent the D_{i+1} resolver IP.

The resolver message packet is designed to contain both the `unknown-list` and the rate of incoming DNS queries targeting the victim—the "DNS query rate." Resolvers keep track of the resolvers to which they have sent DNS query rates and ensure that all resolvers in the `unknown-list` have the rates. That way, the victim's global DNS query rate is updated at the same time as the resolvers build the complete list of resolvers involved in the attack.

E. Complexity

In this section, we characterize the complexity of our communication protocols in terms of the number of resolvers (n) involved in an ADD attack. As discussed in Section III-D, each resolver has access to n number of resolver IP addresses as part of the `unknown-list` at the end of an attack. Resolver D_i sends the IP address of D_{i+1} as well as the DNS query rates of victims to all resolvers in the `unknown-list`. Hence, in the worst case, D_i will send $O(n)$ messages, and the total worst case complexity for n resolvers involved in an attack will be $O(n^2)$. In Section V-F, we further explore the techniques for reducing the complexity to linear and logarithmic polynomials. The network bandwidth consumed by victims and resolvers from exchanging resolver messages will increase linearly with respect to the growing number of resolvers involved in an attack. About 99% of the resolver messages will contain just one IP address of a DNS resolver and a DNS query rate. We expect that such messages will be about 12 bytes, not including any lower-level headers.

F. Detection checks

Each DNS resolver maintains a local information table (LIT) containing DNS queries rate per IP. This table can be maintained as part of a resolver monitoring service and consumes few memory resources. During the attack, the *victim's DNS query rate* is shared across all suspicious resolvers involved in the attack. The resolver's LAD continuously accumulates those rates and performs the threat assessment. The LAD computes threat bandwidth (= accumulated DNS query rate of victim $\times$ amplification factor $\times$ average query size) for the victim, which represents the potential bandwidth of volumetric attack as it builds. Generally, amplification is $\approx 20 - 70$x and query size is ≈ 64 bytes. If the calculated threat bandwidth exceeds a certain threshold, the LAD declares an attack and the DNS resolver blocks any further traffic to the victim. We sometimes refer to a resolver that stops traffic as an activator resolver. Note that the victim could redirect its clients (or DNS traffic) using abused resolvers to 'safe' DNS resolvers.

IV. Performance evaluation

A. Prototype implementation

To demonstrate the feasibility and performance of DRS-ADAM, we implemented a prototype in Python and used it to build a simulation testbed. All of the core DRS-ADAM components, including the local detector and query rate sharing modules deployed on DNS resolvers, and the victim side module that monitors incoming DNS responses and communicates with DNS resolvers, were fully implemented. The DNS resolver message was constructed based on the packet structure shown in Figure 2. Field `Type` is used to identify message type (in our case there is only one kind, i.e., the resolver message) and other fields are self-explanatory. The victim message (used to communicate with resolvers) was constructed using the same packet structure, but the query rate was set to `NULL`. UDP-based communication protocols were implemented as described in Sections III-C and III-D, exchanging those two types of messages.

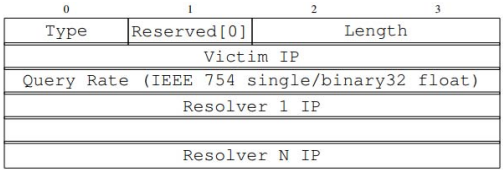


Fig. 2: Packet structure used by DNS resolvers.

the resolver detectors.

B. Methodology

1) Simulation environment: **Hardware/virtual network** We used a DELL T320s server with a 2.4 GHz Intel Core 10 CPU and 16 GB of RAM, running VMware ESXi v5.5 virtualization software. An Ubuntu 14.04.1 LTS virtual machine was deployed on the ESXi server. To build a virtual network, we used network name spaces and virtual Ethernet pairs (veth) mechanisms implemented in Linux OS. Emulated bots, resolvers, and the victim were deployed on the same virtual machine, but each component had its own isolated stack of code, application, and network and was connected using a veth pair.

Emulated topology The tested topology illustrated in Figure 3 was designed to emulate the attack traffic logical topology and the distributed nature of the defense mechanism. Bots and resolvers were organized in clusters; each cluster emulated an autonomous system (AS). The size and the number of clusters for resolvers and bots varied depending on the setting for each test. Each cluster of bots and resolvers was attached to an edge router. These edge routers were connected through transit routers, and the added delay or latency between them was set to emulate an Internet link—aggregation/core.

Attack traffic Each bot in a cluster was programmed to send spoofed TXT queries to a designed cluster of resolvers. Each bot in a cluster sends in parallel using a *thread* to the list of resolvers in the designed cluster. The query is static for each thread to the assigned reflector and amplifier resolver. Once the query for each thread is crafted, the thread will send

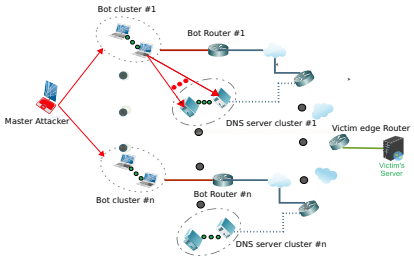


Fig. 3: The emulated topology.

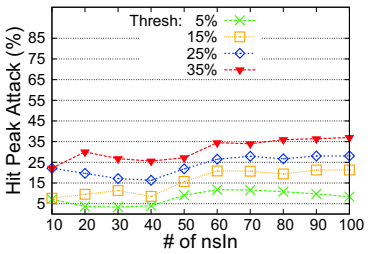


Fig. 4: HPA variation with the number of involved name servers.

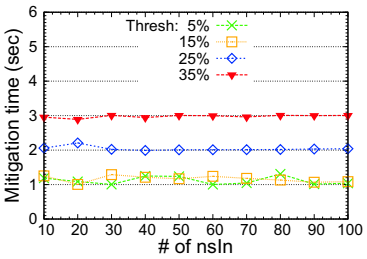


Fig. 5: Mitigation time as a function of the number of involved name servers.

the query—using *raw socket* functionality—in a loop until the end of an attack. This process is similar to the way real-world ADD attacks generate traffic.

The following is a subset of DRS-ADAM test instructions: (i) set up and start the virtual network with the emulated topology; (ii) specify the hosts that will behave as bots or name servers, taking into consideration clustering; (iii) start the implemented resolver detector module (see Section IV-A) on every name server-assigned machine; (iv) monitor name servers' exchange messages; (v) start the implemented victim monitor module at the assigned host; (vi) monitor victim exchange messages and link rates; (vii) master attacker sends commands to run attack traffic on every assigned bot.

In RRL experiments, BIND 9.9.5 was installed on each resolver and RRL was enabled. The victim monitor module was not launched. The resolver response was constructed by adding a zone file to BIND9 where the TXT query response size was configurable. RRL was configured by setting up the `responses-per-second` parameter and the `window` value in '/etc/bind/named.conf.options' file.

Parameters configuration The DRS-ADAM window size was set to 1s. The `responses-per-second` value of RRL was set based on the threshold. The RRL window size was set to the recommended value of 5 seconds. The rate of the attack varied depending on the number of bots; in all the experiments, this number was the same as the number of resolvers. The query rate per resolver was 10 qps. It takes 5 seconds to build a full-size attack, which reflects the worst real-world case—actually, it is in the range of $5-30$ seconds. The duration of the attack was set to 30 seconds.

The scale of these experiments was the largest we could obtain in our testbed environment while guaranteeing the quality and reliability of the results. The performance metrics used in the experiments are: hit peak attack (HPA), DNS attack traffic over time, mitigation time, and system overhead. HPA characterizes the highest point of attack strength, which is defined here as a percentage of the absolute volume of the attack. The mitigation time is the time elapsed from when the attack size at the victim first reaches a minimum and insignificant amount of traffic to be definitely lower than this minimum. Indeed, during the attack's growth phase, the victim will always receive at least one answer or minimum amount of traffic from any newly attacked resolver before that resolver blocks the attack traffic.

C. Results

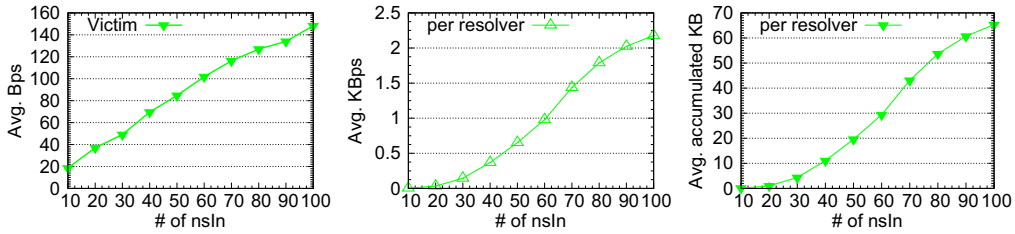
For our experiment, the threshold percent value in the results figures is normalized to the attack's volume. But its real value is actually the *threshold absolute value* given by the victim; this value is a small percentage of the victim's sustainable bandwidth. Indeed, the threshold percentages used in our experiments are very large in comparison to real percentages in order to show how the performance metrics differ with respect to the number of resolvers. This size also gives us more confidence in results when scaling up. In actuality, the *threshold absolute value* will be much smaller than 5% of the attack's size—in the current attack trends, victim bandwidth is usually in the range of $0.03-16\%$ of attack's size and maximum DNS traffic is less than 5% of the victim's sustainable bandwidth. In our simulation, we increased the number of resolvers while keeping the DNS query rate received by resolvers constant. Figure 4, Figure 5, and Figure 6 are plotted for different threshold values.

1) Hit peak attack rate: Figure 4 shows the HPA metric with a varying number of name servers ($nsIn$) for different threshold settings. It demonstrates the impact of an ADD attack on a victim with DRS-ADAM system in place. We can observe that the peak attack traffic at the victim fluctuates with just small variations around the *threshold absolute value* (HPA is close to the threshold percentage) as we increase the number of resolvers; this is clear after 50 resolvers. As a result, inbound attack traffic at the victim will never reach its sustainable bandwidth; for instance, if the *threshold absolute value* is 5% of the victim's sustainable bandwidth, the peak attack traffic at the victim will only be around 5% of the latter bandwidth when DRS-ADAM is in use.

This result implies that DRS-ADAM is *scalable* with respect to resolvers involved in attacks and suitable for mitigating large DNS amplification attacks without disrupting a victim's normal services.

Due to the limited computing resources we had in our experiment setup, we were not able to emulate more than 100 DNS resolvers. However, we further discuss theoretical justifications for the scalability of DRS-ADAM in section V-F.

In fact, DRS-ADAM resolvers are exchanging messages and aware in real-time of the actual global rate of the attack in parallel with the attack growth. The DRS-ADAM messages from the activator resolver—used by the attacker at the threshold growth level—activates previously attacked resolvers to



(a) (b) (c)

Fig. 6: System Workload

drop the spoofed queries received from the attacker, since they already know the past attack's global rate. HPA depends on this delay of activation from when the attack size reaches the threshold to be blocked by all previously attacked resolvers. HPA is close to the threshold percent value as this activation delay is much smaller than the attack's double growth speed after hitting the threshold; the growth phase time of the attack occurs in seconds, usually around $20 - 30$ seconds. The delay time until message receipt from the activator resolver is measured in ms for all previous attacked resolvers; the message delivery time and its transmission delay (including 'build-out' DRS-ADAM packet delay; it was around $0.4\mu s$ on our platform) measured in ms and μs (for any network connection), respectively.

The HPA is slightly sensitive to the number of $nsIn$ as the time difference between when two previously attacked resolvers enter the blocked state of the attack traffic is measured in μs. Note that a victim needs to receive only one mismatch DNS answer—i.e., inbound answer without a matching outbound query—to start communicating with the involved resolvers (see Section III-C).

2) Mitigation time: DRS-ADAM is a quick response system that can mitigate ADD attacks within a few seconds, as shown in Figure 5. We expect even shorter mitigation times in the real-world, as the threshold will be much smaller while the attack growth speed is the same as in the test experiments. Also, mitigation time does not vary with respect to the number of resolvers for a fixed threshold, which demonstrates the scalability of DRS-ADAM for mitigating real-world attacks.

The mitigation time is only slightly sensitive to the number of resolvers as it depends mainly on the following two factors that are practically independent of the number of resolvers. (i) The first factor is the time it takes an attack to reach the threshold percent value of its total size. The defined time for any fixed threshold percent value is independent of the number of resolvers—the time taken for an attack to reach its maximum size is fixed and similar to real cases, independent of the number of resolvers. (ii) The second factor is how quickly DRS-ADAM resolvers block their local attack traffic relative to the growth speed of the attack after the attack has reached the threshold size. This blocking speed as discussed for Figure 4 is much faster, and we can apply the same reasoning about HPA insensitivity to the number of resolvers.

The time taken for the attack to reach the four thresholds of $5, 15, 25, 35$ percents are $0.25, 0.75, 1.25,$ and 1.75 seconds, respectively. Our mitigation times included those build up times, so there is half a second difference in mitigation time between any threshold percentage value, regardless of DRS-ADAM mitigation time performance at different thresholds.

When DRS-ADAM is used, the victim's inbound attack traffic grows in seconds under an ADD attack, hits a peak value close to the *threshold absolute value*, and then declines substantially after this highest point. The peak is short-lived, measured in ms, and the decline occurs within a few seconds (< 1.25s for any threshold and number of resolvers).

3) Bandwidth and memory consumption: A victim's outbound bandwidth consumption increases linearly in bytes per second with respect to number of resolvers involved in an ADD attack, as shown in Figure 6a. During an attack, a range of tens of thousands of resolvers corresponds to victim host bandwidth consumption of a few KBps. This consumption is negligible in comparison with the MBps or GBps bandwidth available to a victim, and thus, does not affect its system workload in any way.

Similarly, the average bandwidth consumption for each resolver increases linearly with the number of resolvers involved in an attack, as shown in Figure 6b. If we again scale the number of resolvers to tens of thousands, the worst case corresponds to a few hundred KBps bandwidth consumption per resolver. This amount of consumption is manageable and does not disrupt the main activities of a DNS resolver. Figure 6c represents the total bandwidth consumption per resolver during the whole attack period. For real-world cases like spamhaus, this consumption corresponds to a few MB of bandwidth, which is quite manageable.

Each resolver maintains a state table of query arrival time-stamps in order to calculate its local attack rate. The maximum size of this table is the product of maximum queries per second and window size. Assuming 2 bytes per time-stamp, 600 qps rate, and a window size of 1 sec, a resolver consumes 1200 bytes of its memory.

4) Comparison with RRL : To show mitigation capabilities against highly distributed ADD attacks, this comparative experiment with RRL used a fixed attack size regardless of the number of resolvers. The *absolute threshold value* stayed the same for any threshold percentage, while the number of resolvers increased. As a result, the attack query rate per resolver decreased. To the best of our knowledge, this work

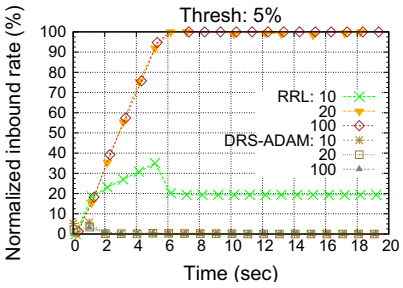


Fig. 7: Normalized inbound rate over time of the victim's router.

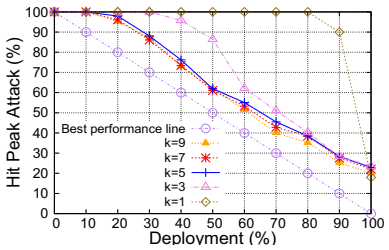


Fig. 8: HPA variation for partial deployment of DRS-ADAM with 100 resolvers and 15% threshold.

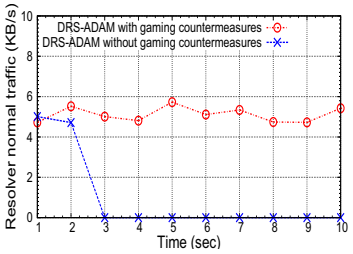


Fig. 9: Manipulating (gaming) DRS-ADAM and observed resolver normal traffic.

presents the first experimental evaluation of BIND9's RRL feature in a distributed setting with multiple DNS resolvers; to date, RRL has been tested only on a single DNS server. Figure 7 depicts the normalized inbound rate (NIR) observed at the victim's router interface for different numbers of resolvers. The threshold used is 5%. NIR is the real-time attack rate percentage of the peak expected attack's volume at the victim (the total attack volume after the growth phase when no mitigation is in place). The `responses-per-second` value of RRL is set to 10 based on the 5% threshold.

When the number of resolvers is greater than or equal to 20, the RRL *rate-limit* response is not activated during the attack, and the victim host faces downtime. When the ADD attack becomes well-distributed, using more resolvers at the same attack size, it can bypass the RRL detection mechanism when the attack query rate per resolver becomes less than or equal to the per-second limit of RRL (10 qps in the 20 resolvers' case). Thus, RRL will have little or no impact on highly-distributed ADD attacks. Even when RRL can detect an attack in the case of 10 resolvers, it cannot stop the attack traffic before the HPA hits 34.8% for the 5% threshold value. This trait shows that high-scale attacks might still be effective even if they are detected and rate-limited. In contrast, DRS-ADAM stops all attack traffic in less than two seconds and contains HPA at around 5%. The NIR of an RRL for 10 resolvers rose to 19.3% after one second, gradually increased by 3.8%, and finally settled at 19.3% after 6 seconds. In the first second (Second 1), two Group 1 resolvers are each attacked at a rate of 20 qps; the RRL *rate-limit* response is activated for a small fraction of queries (one query in the first second). This response contributes approximately 19.3% to the NIR. Note that an attack's start second is different from the first RRL counting second. Here's what happens in the test: in the first RRL counting second, the attacker sends 9 queries, while in Second 2, the attacker sends 11 queries; because the `responses-per-second` limit is 10, the second RRL account becomes negative at each resolver only at the end of Second 2. For every received query, the RRL counter is decremented by one, and at the start of every counting second it is increased by the `responses-per-second` value and is limited to that value. At the start of Second 2, the Group 1 resolvers' RLL *rate-limit* response is negative and activated to send a truncated response to every other incoming query (BIND9 `slip` default value is 2). This truncated response (used for false positive mitigation) is the same size as the query and guarantees that the query is not amplified. Therefore, Group 1 adds around 3.8% ($\approx$ 20 unamplified replies—out of 200 amplified answers—that the same size as the queries) to the NIR in the full second. Also, at the start of Second 2, two new resolvers (Group 2) are attacked. Their RRL *rate-limit* response will be activated for only a few queries (as with Group 1 during Second 1) in this second and thus contribute approximately 19.3% to the NIR. In Second 3, Group 2's RRL account is negative and starts to behave like Group 1. Together, Groups 1 and 2 contribute 7.6% to the NIR and Group 3 contributes 19.3%. In Second 4, the first three groups contribute 11.4% to the NIR and Group 4 contributes 19.3%. In Second 4, the first four groups contribute 15.2% to the NIR and Group 5 contributes 19.3%. In Second 6, all the groups contribute 19.3% to the NIR and stays the same (RRL account at any resolver will remain negative) until the end of the attack.

5) Partial deployment of DRS-ADAM: In this section, we evaluate the effectiveness of DRS-ADAM when it is *partially deployed* on a set of resolvers. DRS-ADAM is designed to propagate the requisite information sequentially from one resolver to another; i.e., by forming a chain of resolvers to reduce the message exchange complexity. If DRS-ADAM is not deployed on any one of the resolvers involved in an ADD attack, then it will break that chain of information propagation and form disjointed groups of resolvers that cannot exchange query rates with other groups. To avoid that situation, we enhanced our original victim and DNS resolver communication protocol described in Section III-C as shown below.

$D_i \longrightarrow V \quad : SDR_i$	Victim V receives a spoofed DNS response from D_i for the first time.
$V \longrightarrow D_{i-k} : \{DIP_i, .., DIP_{i-k+1}\}, threshold$	Victim V forwards threshold and IP address of $\{DIP_i, DIP_{i-1}, .., DIP_{i-k+1}\}$ to D_{i-k}.

Resolvers can now receive multiple DNS resolver IPs from the victim monitor module based on the value k. This flexibility in k helps predecessor resolvers to be connected with successor resolvers when the chain is broken by a resolver that is not equipped with DRS-ADAM.

Figure 8 shows HPA for different percentages of resolvers equipped with DRS-ADAM. Resolvers that are equipped with DRS-ADAM were randomly selected from a set of 100 resolvers. In particular deployment scenarios, there will always be a minimum HPA value, since resolvers that are not equipped with DRS-ADAM cannot stop an ADD attack. The best performance line in the graph represents this minimum HPA that will be observed when a *theoretically best-possible*, resolver-based (both local and distributed) solution is deployed.

By increasing k, the DRS-ADAM performance improved, and eventually when $k \geq 5$, gave close to that best theoretical performance. The message complexity for the victim would now be $O(kn)$, increasing linearly with the value of k. For a small constant k value; however, it is safe to expect the complexity to remain at $O(n)$.

6) Manipulating DRS-ADAM and defending against UDP spoofing : The attacker can use various methods to manipulate (or game) our DRS-ADAM system by exploiting its communication protocol. In one scenario, an attacker can try to falsify the DNS query rates or *unknown-list* circulating among DNS resolvers by forging UDP packets. The attacker can send a resolver packet of less than the actual DNS query rate (even 0). DRS-ADAM solves this problem by considering the packet with the maximum DNS query rate but limited to an upper rate bound. This received attack rate limit prevents a bogus message with a large rate from blocking normal DNS traffic to a client. The limit is set to the maximum attack rate per resolver of the real-world largest DNS amplification attack [14].

An attacker can also send a huge garbage `unknown-list` to increase a DNS resolver's communication overhead. DRS-ADAM cross validates list by obtaining the DNS query rate for a few random resolvers; the rate should be greater than 0.

We implemented both these logics in our DRS-ADAM system and simulated a manipulation attack by constantly generating forged UDP packets (with an interval of 100 ms) that each carried either a lower DNS query rate or a huge garbage list. Our DRS-ADAM successfully dropped all such packets and our results remained same as shown in Figures 4,5.

Another scenario exploits DRS-ADAM to block normal DNS traffic even when the victim is not under ADD attack. Here, the attacker sends forged packets of DNS query rates. DRS-ADAM cross validates query rates from randomly selected resolvers just before blocking the victim services. The attacker can further attempt confusion by sending responses from all resolvers. DRS-ADAM can identify the type of attack as it does not expect packets from all resolvers. For this experiment, we sent forged DNS query rates along with normal DNS queries from the victim to a resolver.

Figure 9 shows that the DRS-ADAM manipulation prevention system successfully continued serving DNS queries for the victim even when an attacker tried to exploit it. Furthermore, we strengthened our cross validation technique by attaching a nonce to resolver messages and expecting the same nonce in return. Additionally, each DNS resolver considers the lowest threshold retrieved from the 'victim' messages but also limited to a *lower threshold value*. This measure eliminates threshold setting manipulation. The *lower threshold value* is fixed on the DNS resolver side. A conservative, lower threshold value of 5Mbps could also be used, as most network connections are measured in 100Mbps, 1Gbps or 10Gbps.

When these prevention measures are in place, an attacker could still succeed by using multiple machines and broadcasting fake rate messages to block normal DNS traffic to a client. A DNS resolver prevents this situation by checking whether its local 'victim' rate is smaller than the lower bound of the 99.999% confidence interval of the received attack rates. When this measure is in place, an attacker desiring to block normal DNS traffic to a client by DRS-ADAM manipulation will need more machines than to DDoS directly to the target host.

V. Discussion

This section examines what it means to deploy DRS-ADAM in the real world and discuss the scalability and optimization techniques for reducing communication overhead.

A. Satisfying design goals

Our first design goal, as stated in Section III-A, was to contain ADD attacks under victim-specified acceptable bandwidth threshold values. Our HPA results (see Section IV-C1), showed that DRS-ADAM is capable of keeping ADD attacks below different ranges of threshold values at all times, and stopping ADD attacks completely. It also surpassed RRL in containing and stopping ADD attacks, as is evident from section IV-C4. Since DRS-ADAM is purely a software-based solution that requires just a small upgrade on DNS resolvers and victim firewalls, it satisfies our second goal of achieving high deployability and scalability; we discuss our deployment strategies in the next section. Lastly, all DRS-ADAM communication protocols are UDP based, and its architecture does not rely on any cryptographic key based authentication scheme, which satisfies our last two goals of developing a purely UDP based solution and avoiding the use of expensive PKI.

B. Economical incentives for network operators to deploy DRS-ADAM

With DRS-ADAM installed, in a real ADD attack scenario like spamhaus [14], all resolvers involved in the attack will consume bandwidth in *KBps* for few seconds during the process of sharing query rates before stopping the attack completely (see section IV-C3). This amount of bandwidth consumed to facilitate DRS-ADAM communications during the initial stages of the attack is infinitesimal compared to the bandwidth that would be consumed if the attack runs (i.e., without DRS-ADAM) at $\sim 10 - 300$ Gbps and lasted for a week. Such network bandwidth costs can be avoided with DRS-ADAM in place, and result in considerable yearly cost savings for network operators. In the example scenario described in Section II, an ADD attack can potentially incur network costs up to $\$1,800,000$ per month, which would be shared among the network operators involved in an attack. With the rapidly increasing size and frequency of ADD

attacks, we expect such bandwidth consumption costs that network operators would have to pay will also increase exponentially over time. Hence, in long run, network operators (that manage DNS resolvers) will benefit immensely in terms of avoiding unwanted bandwidth consumption costs by installing DRS-ADAM on their resolvers and participating in the global detection efforts—saving tens of thousands of dollars per year.

C. Deployment strategies

Our solution can be thought of as a pair of small software packages. The first package needs to be installed on DNS resolvers; the second package is installed on the victim side (e.g., on a firewall). The victim package should be small (our prototype had just 250 lines of code) and should not require frequent updates. These installation packages need to be deployed on top of existing DNS software and protection mechanisms. All existing DNS software such as BIND [3] already provides real-time statistics about DNS queries and responses that our solution can exploit. We can thus deploy our solution as a single upgrade to existing DNS software in the same manner as RRL was deployed on BIND software and pushed out so that all DNS resolvers support it.

Our deployment strategy faces fewer challenges than do domain name system security extensions (DNSSEC) [16] used for securing DNS systems, which rely on digitally signed records. DNSSEC deployment is much more complicated owing to complex key management and backward-compatibility. In contrast, DRS-ADAM needs only a single software update and does not add any complex key management or hardware upgrade overhead. In Section IV-C5, we showed that, even in partial deployment scenarios, DRS-ADAM can perform close to the best theoretical possible DNS resolver based solution.

D. Returning to the normal state

Resuming normal traffic for the victim occurs in the same way as blocking. Resolvers communicate at regular intervals after blocking the attack to evaluate the threat bandwidth until it falls below the threshold. In current practice, it is reasonable for this value to be in the range of 10-30 minutes, as attackers tend to give up quickly once DNS amplification is not successful (though, of course, this might change).

E. Effectiveness against highly distributed attacks

To make an ADD attack as difficult as possible to detect, an attacker could try to perform a highly distributed attack by involving a huge number of DNS resolvers and sending a very small number of queries (e.g., one or two) to each resolver. It would be extremely difficult for any statistical model that makes decisions based on the locally observed query rates, including RRL, to detect and stop this kind of attack.

DRS-ADAM, however, would still be effective against such an attack because it primarily focuses on the *total accumulated DNS query rates*, which ensures that it is consistently effective regardless of the size of the local query rates observed by DNS resolvers. Moreover, as stated earlier, the capability of DRS-ADAM to work with dynamic threshold values (see Section III-C) ensures that attacks are contained within the victim-specified threshold bandwidth regardless of the size of an attack or the number of resolvers involved in an attack.

F. Bandwidth optimization and DRS-ADAM scalability

In our current communication protocol, each resolver forwards the `unknown-list` and query rate to a new incoming resolver and back-propagates the new resolver IP and its own DNS query rate to all previous known resolvers. It is also possible to forward only the accumulated DNS rates. Once the accumulated DNS rates cross the threshold, that activator resolver can back propagate its decision and inform other resolvers to stop the traffic. In this case, the activator resolver sends $O(n)$ messages and all other resolvers send only one. To protect trust among resolvers, the activator resolver can send the individual DNS rates of all resolvers. Each resolver can then verify by randomly sampling the list and validating the rates for the sample. Rather than just propagating accumulated DNS rates, a resolver also needs to provide the individual DNS rates of all previous resolvers. This practice reduces the number of messages to a single message of increasing size. This method is a possible way to optimize communication by balancing the complexity of messages against the robustness.

To further increase the scalability of DRS-ADAM, we can include a known, provably-effective method for diluting communication overhead among the resolvers while preserving accurate estimates of the network-wide, cumulative query rate: a Gossip-based Push-Sum Protocol (GBPS). GBPS can compute a cumulative query rate over all node-values quickly and accurately and is completely robust to network size, ensuring scalability [17]. These methods have also been tailored for ongoing network dynamics with continuous calculation for real-time updates [18], which may allow easing of the assumption of constant query rates per resolver (see Section III-B).

G. Generalization

The proposed solution can be generalized to mitigate any amplification attacks possessing characteristics of DDoS attacks and UDP-based communication. For example, an NTP amplification attack [19] is similar to DNS amplification but uses NTP servers instead of DNS servers. The attackers generate large numbers of spoofed responses targeted at a victim. These queries are then amplified using the MONLIST command and sent to the victim by NTP servers with an amplification factor of about 200. Recently, many NTP amplification incidents have been reported, including the largest ever DDoS attack of 400 Gbps. In our proposed solution, NTP amplification can be easily detected by replacing the DNS resolver component with an NTP server and adjusting the threshold accordingly. This solution can be a software package installed on both the NTP server and the client for protection against NTP amplification attacks.

VI. Related work

The majority of the proposed solutions for mitigating DNS-based DDoS attacks are deployed at the victim level. As discussed in Section II, such solutions do not address the

root cause of the attacks. Sun et al. [13] and Kambourakis et al. [4] use a basic approach of detecting DNS-based DDoS by keeping a one-to-one mapping of DNS query and responses; Jose et al. [20] propose a detection method based on DNS log monitoring with Hadoop. Rastegari et al. [12] and Deshpande et al. [21] proposed different versions of neural networks and probabilistic model to classify attacks. Such approaches are effective in detecting the attack but cannot avoid flooding the victim's bandwidth with DNS responses. For a huge attack, this approach results in consumption of victim resources and failure of the system. Other approaches use machine learning algorithms for classifying the ADD attacks.

ADD relies on the ability to spoof IP addresses, so source address validation solutions can be effective. These solutions include authentication-based methods [22], traceback based methods [23], and filtering-based methods [24], [25]. Deployment of such solutions is practically limited because they require fundamental changes in Internet infrastructure. Ingress filtering allows routers to drop packets with source IPs that do not match any entry in the FIB forwarding table. The table lookup time adds an important delay on a router's forwarding time, which is unacceptable on high-speed links. Note that amplified answers from DNS resolvers will not be dropped by ingress filtering as they have a legitimate DNS source IP address, thus this filtering requires a high fraction of deployment. DNS-Guard [11] addresses the DNS amplification attack at the root cause by developing a mechanism for detecting spoof DNS queries for DNS servers. It uses a cookie-embedded query and response to authenticate the source. However, this approach introduces a fundamental delay in DNS response time and requires additional servers running in between hosts and root servers, which is hard to achieve in practice.

VII. Conclusions

Existing mitigation techniques for ADD attacks cannot prevent a victim's network bandwidth from being fully consumed even with timely detection of the attack. Our novel, distributed mitigation system, DRS-ADAM, can quickly detect ADD attacks and fully stop them well before they pose a significant threat to victims. To detect an ADD attack, DRS-ADAM utilizes the DNS resolvers that are exploited in the attack and does not require any additional, dedicated hardware. DNS resolvers share the local DNS query rates with each other, facilitating quick and accurate computation of accumulated query rates.

Our simulation results show that DRS-ADAM is scalable for both mitigation times and hit peak attack rates. ADD attacks will always be contained close to victim's acceptable thresholds, regardless of the number of DNS resolvers involved in an attack, and mitigated within a few seconds. DRS-ADAM is robust against manipulation scenarios such as falsifying the DNS query rates shared among DNS resolvers (with UPD packet forging). As part of future work, we plan to further simulate manipulation attacks, and evaluate the robustness of DRS-ADAM against such attacks.

Acknowledgment

This research was supported by the US Department of Homeland Security sponsored under the Air Force Research Laboratory (AFRL) agreement number FA8750-12-2-0232.

References

[1] T. Rozekrans, M. Mekking, and J. de Koning, "Defending against dns reflection amplification attacks," *University of Amsterdam, Technical Report*, 2013.

[2] "Open Resolver Project," http://openresolverproject.org/breakdown.cgi.

[3] "BIND," https://www.isc.org/downloads/bind/.

[4] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "A Fair Solution to DNS Amplification Attacks," *International Annual Workshop on Digital Forensics and Incident Analysis*, 2007.

[5] "Incapsula Survey : What DDoS Attacks Really Cost Businesses," http://lp.incapsula.com/rs/incapsulainc/images/eBook%20-%20DDoS%20Impact%20Survey.pdf.

[6] "Akamai's state of the internet and security," http://media.scmagazine.com/documents/144/q2_2015_soti_security_report_-_35820.pdf, 2015.

[7] "Introduction to dns ddos amplification attacks," http://nominum.com/wp-content/uploads/Introduction-to-DNS-Based-DDoS-Attacks.pdf.

[8] "The open internet: a platform for growth," http://www.plumconsulting.co.uk/pdfs/Plum_Oct11_The_open_internet_-_a_platform_for_growth.pdf, Plumconsulting.

[9] "Verisign distributed denial of service trend report," https://www.verisign.com/assets/report-ddos-trends-Q12015_en_IN.pdf.

[10] "How Many DDoS Attacks Happen Each Day?" http://www.globaldots.com/many-ddos-attacks-happen-day/.

[11] F. G., J. Chen, and T. Chiueh, "Spoof Detection for Preventing DoS Attacks against DNS Servers," *International Workshop of Software-Defined Data Communications and Storage*, 2006.

[12] S. Rastegari, M. I. Saripan, and M. A. Rasid, "Detection of Denial of Service Attacks against Domain Name System Using Machine Learning Classifiers," *International Journal of Computer Science*, 2009.

[13] C. Sun, B. Liu, and L. Shi, "Efficient and Low-Cost Hardware Defense Against DNS Amplification Attacks," *GLOBECOM*, 2008.

[14] "The DDoS that Almost Broke the Internet - March 2013," http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/, CloudFare.

[15] R. Pangy, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A first look at modern enterprise traffic," *Internet Measurement Conference*, 2005.

[16] DNSSEC, "Dns security extensions securing the domain name system," http://www.dnssec.net/.

[17] D. kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," *IEEE Symposium on Foundations of Computer Science*, 2003.

[18] A. M. M. Jelasity and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, 2005.

[19] "US-Cert Alert (TA14-013A) NTP Amplification Attacks," https://www.us-cert.gov/ncas/alerts/TA14-013A, US-Cert, 2014.

[20] A. S. Jose and A. Binu, "Automatic detection and rectification of dns reflection amplification attacks with hadoop mapreduce and chukwa," *International Conference on Advances in Computing and Communications*, 2014.

[21] T. Deshpande, P. Katsaros, S. Basagiannis, and S. A. Smolka, "Formal Analysis of the DNS Bandwidth Amplification Attack and its Countermeasures Using Probabilistic Model Checking ," *International Symposium on High-Assurance Systems Engineering*, 2011.

[22] A. Bremler-bar and H. Levy, "Spoofing prevention method," *IEEE INFOCOM*, 2005.

[23] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-Based IP Traceback," *ACM SIGCOMM*, 2001.

[24] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2267," 2000.

[25] Y. Katsurai, Y. Nakamura, and O. Takahashi, "A proposal of a countermeasure method against dns amplification attacks using distributed filtering by traffic route changing," *International Workshop on Informatics*, 2015.