Dealing with DNS Amplification Attacks using Response Rate Limiting

Pedro Madeira
Instituto Superior Tecnico, Universidade de Lisboa

***Abstract*—The Domain Name System (DNS) provides one of the most fundamental functionalities of the Internet as we know it, which is resolving names into IP addresses and vice-versa. Unfortunately it relies heavily on the UDP transport protocol making it vulnerable to DDoS reflection attacks. It also allows attackers to amplify their malicious traffic through DNS servers. Small queries can generate responses many times their size. Combining these two factors made DNS one of the most abused systems by DDoS attacks, that have also been increasing in scale in recent years. As the entity that oversees the critical function of the Top-Level-Domain corresponding to Portugal (.pt), DNS.pt has the responsibility to avoid causing a major economical and social impact in its partners, not only damaging other businesses but also their own reputation. A recently introduced mechanism called Response Rate Limiting has shown to be effective in mitigating these kind of attacks. This work studies its application for the Top-Level Domain of Portugal (.pt), managed by DNS.pt.**

I. Introduction

One of the most important systems that make up the Internet as we know it today is the Domain Name System (DNS). It provides one of the most fundamental functions of the Internet which is resolution of domain names into IP addresses and vice versa (whether they are IPv4 or IPv6). In its turn, DNS relies on the UDP transport protocol, making it particularly vulnerable to Distributed Denial of Service (DDoS) attacks.

A DNS server may not only be the target of a DDoS attack, but also the accomplice of one. Since UDP is stateless, there is no validation of the source address of packets in the network, making it impossible for a DNS server to distinguish a forged request from a legitimate one allowing attackers to send forged DNS requests spoofed with their victims address. This results in a reflection of malicious traffic on DNS servers. This could be eliminated by enabling source-based filtering techniques on Internet Service Providers (ISP) networks. Unfortunately, deployment of these is not being widely adopted which makes them ineffective.

Additionally, the DNS service allows attackers to amplify their malicious traffic since simple queries to the DNS service can yield much larger answers. For instance, a 30 bytes long query message can generate an answer over 3000 bytes.

These two factors, reflection and amplification, make DNS an attractive system to carry attacks that may not even target DNS servers themselves but other companies and organizations. When this happens, the victims see the DNS service as their offenders. DNS.PT Association is the national registry for the Internets Top-Level Domain (TLD) corresponding to Portugal (.pt). They oversee management, assignments, operation and maintenance of the domain registries. Due to DNS.PT's importance in securing essential critical functions at economic and social levels, disturbance of its operation can have a major impact in the wellbeing of the country. It is therefore its responsibility to conduct studies and develop solutions that promote the protection of their services.

There are already ways of reducing the impact of DDoS attacks that abuse DNS servers, although most of them suffer from deployment issues or may have an impact for legitimate users. A recent mechanism called Response Rate Limiting (RRL), deployed inside DNS servers themselves, was specifically created to deal with the issues of reflection and amplification. There has not been a lot of feedback on how the mechanism behaves and impacts the normal operation of a DNS service. Because of that, deploying it in such an important service as the one provided by a TLD should be preceded by a careful study on how to configure it to avoid harming legitimate users.

The main goal of this work is to study the usage of the RRL feature on the DNS service provided by DNS.pt as a way of mitigating DDoS Attacks that may use DNS servers as reflectors and amplifiers. In doing so, we should achieve a better understanding of how the feature will work in a TLD server scenario.

II. Background

A. Distributed Denial of Service (DDoS)

One of the most common types of DoS is the DDoS in which multiple attacking agents cooperate to cause excessive load to a victims service or network. This kind of attack usually takes advantage of botnets composed of multiple compromised hosts running one or more bots controlled by one master program called Command and Control (C&C) program. Figure 1 illustrates this kind of attack. The attacker, through the C&C program, sends instructions to the compromised hosts running the bots to perform tasks that actually consist of the attack. This not only hides the real attacker but allows it to scale the impact on its victim(s). In fact, recent DDoS attacks have reached traffic volumes close to 1 Tbps [1] with hundreds of thousands of devices participating simultaneously in them [2].

B. DNS Amplification and Reflection

A DNS Amplification and Reflection attack is a type of DDoS attack that takes advantage of DNS servers and protocol

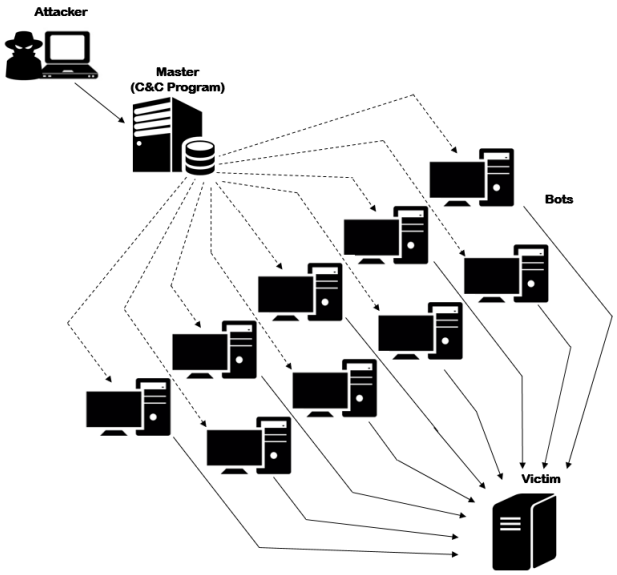

Fig. 1: A Distributed Denial of Service Attack illustration.

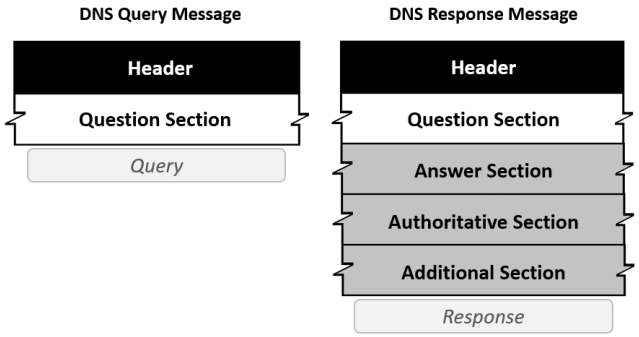

Fig. 2: DNS protocol messages format [4]: query and response.

in order to flood a victim's system or network with a large volume of traffic [3].

This type of attack is based on the DNS protocol, which is a request-response protocol relying heavily on UDP [4]. Unlike TCP, there is no handshake between the system that sends a request to a server. This means the server will immediately send a response to the source IP address that comes in the request message. Attackers exploit this by spoofing the source IP address in order to make the DNS server send the response to their targets.

Reflection itself is not the most attractive exploit for attackers. Because DNS response messages are almost always larger than their requests (see Figure 2), a DNS server acts as a traffic volume amplifier. Response messages carry the same question section as the request message plus all the records in the answer, authority and additional sections. The amplification factor is given by the ratio between the size of the response message and the request message.

Not including IP and UDP headers, a request message usually has a few dozen bytes depending on the size of the QNAME field. A small response message with only one resource record in each of the three previously mentioned sections can double or triple the size of the request message. However, a study by MacFarland et al., in which resource records of nine TLDs were queried, shows that much greater amplification can be easily obtained [5].

Due to this amplification effect, attackers are able to generate enormous amounts of traffic towards their victims; the spoofing of the source IP address prevents the attackers from being blacklisted because their real IP address is unknown; and the victims may not be able to blacklist the IP address of the DNS servers because it eliminates all legitimate DNS resolutions [5], [6].

1) Query types and amplification: One aspect attackers consider in order to greatly increase the volume of traffic sent to their victims is the types of query they send to the servers. A particular query type, the `ANY` query, asks the server to return all known resource records associated with a name. For instance, issuing `ANY` queries for base domain names commonly returns `SOA`, `NS` and `MX` records along with A records associated with the host names [7]. Because of this it is very common to see attackers using the `ANY` query type. In fact, in most of the observed attacks of this type the spoofed queries sent by attackers were of type `ANY` [3].

However, attacks continuously increase in sophistication and there are other ways to achieve higher amplification ratios. The TXT resource record, for example, allows arbitrary text to be stored using the DNS [8]. This had led to a recently observed tendency of attackers crafting TXT records to increase the impact of their attacks [9].

2) Effect of EDNS and DNSSEC: DNSSEC introduces additional resource records and most of them can be very large due to carrying digital signatures. Since it implicitly requires the use of EDNS, signaling support for DNSSEC allows the usage of the UDP payload extension. DNSSEC-signed domains may be used to achieve even higher amplification ratios. DNSSEC enabled requests may yield responses to query messages that are over 50 times larger than responses without DNSSEC records [10].

III. DDoS Prevention, Detection and Mitigation

There are two major types of defense mechanism against DDoS attacks: filtering and rate limiting. Although similar, since both employ some sort of packet/message dropping, these two types of defense usually work at different points or layers.

A. Filtering-based mechanisms

One way of preventing DDoS attacks is to perform filtering of network traffic at some point before it reaches the intended victim. The following are some of the filtering mechanisms mainly focused on dealing with traffic amplification and/or reflection.

1) Network Ingress Filtering: Network Ingress Filtering [11] is a very effective way of dealing with the key aspect of reflection attacks which is source IP address spoofing. Also known as BCP38 [11], it allows routers to check if the

source address in a packet is a valid address of the network it is coming from. Internet Service Providers (ISP) customers receive IP addresses from a certain range, e.g., 1.0.0.0/16. If the ISP's router receives packets from the customers with source addresses outside that range, e.g., 2.0.0.1, it should drop the packets. Preventing source address spoofing would not allow reflection attacks to happen. Amplification would still happen due to protocol nature but attackers would not be able to direct the amplified traffic to their victims unless their victims were on the same subnet as them.

2) Route-Based Packet Filtering (RPF): Border routers are the Internet topology connectors, connecting the various routing domains (AS) that compose the Internet with each other. Routes are propagated throughout these using BGP. As the name implies, RPF performs filtering for flooding spoof traffic at these Internet border routers. RPF does this by having access to the BGP routing topology.

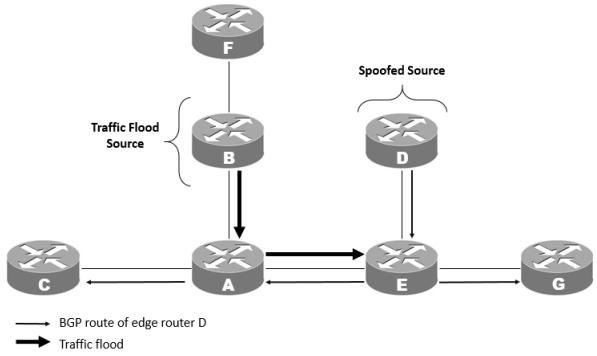

Fig. 3: Example of multiple routing domains, or ASes, connected.

In the routing topology of Figure 3, consider a host inside AS B is sending traffic to a host in AS E with spoofed source address from AS D. A border router in AS A, aware of the topology, upon receiving those packets would recognize them as spoofed traffic because traffic from AS D should not come from the connection with AS B. It is essentially a sparse variant of ingress filtering. Instead of filtering in customer edge routers, filtering only needs to be done in AS border routers which are much fewer than the former.

RPF can be effective, but its major setback is that its effectiveness depends heavily in a deployment strategy, not only in number of AS that do it but also which [12], [13]. The effect of these source-address based mechanisms depends heavily on universal deployment. If only a few AS deploy this mechanism, attackers may simply choose other networks or hosts that do not stop spoofed traffic. It requires a global combined effort for these mechanism to be successful in stopping DDoS attacks that use source IP address spoofing.

3) Dropping ANY queries: ANY type queries have the highest amplification potential since answers to it return all resource records associated with the domain name. Attackers frequently use this query type to increase the amplification ratio, making it the most used query type in observed DNS Amplification attacks [3]. Based on the fact that most clients do not rely on these queries, a very simple way of reducing the impact of amplification attacks would be to have a firewall drop all requests of type ANY before they reach the name servers. However, there are at least two issues with this approach. First, it would probably result in false positives, dropping legitimate queries as well. Second, and most importantly, is that with the rise of DNSSEC and other resource records like RRSIG, DNSKEY or TXT that cause large amplifications, attackers could simply switch to DNS queries of those types. An approach that consists in having a firewall drop malicious traffic would require a much higher level of sophistication to deal with equally sophisticated attacks.

4) D-WARD: Mirkovic et al. [14], [15] presented a scheme which aims to detect DDoS attack traffic by monitoring inbound and outbound traffic of a source network and comparing traffic information with predefined normal flow models. Malicious traffic flows are detected and filtered when they don't match the models. For instance, a TCP packet should always be acknowledged. A normal TCP traffic flow would not exceed a certain ratio of sent and received packets for one peer. Calibration for each kind of traffic reduces the chance of false positives.

D-WARD may be effective at stopping unusual traffic flows but it may not detect traffic that although being malicious and part of a reflection-based DDoS, still follows a normal pattern of a UDP traffic flow. Massive DDoS attacks, carried out by a large number of bots could easily avoid a solution that tries to detect abnormalities in traffic flows close to the source. Large volumes of traffic in DDoS reflection attacks are mostly achieved by a large number of bots sending traffic at reasonable rates instead of a few sending large volumes that could easily be identified as malicious.

5) DAAD: DNS Amplification Attack Detector (DAAD) [16] is based on the idea that a DNS request should create a corresponding response and any other responses that do not pair with a previous request should be marked as suspicious. It collects DNS requests and responses using the IPtraf[1] tool. Once an IP address has raised enough suspicion due to unrequested DNS responses, DAAD blocks that IP address. This proposal aims to mitigate the effects of DDoS flooding attacks on the victim side, most likely at the edge of its network. It is an effective solution at stopping unrequested traffic. From a DNS server operator perspective it becomes useless because it not only the servers receive the requests but the responses usually go out through the same path the request came.

B. Rate-Limiting

Another approach to deal with malicious or abnormal traffic, specifically traffic sent to DNS servers, is to impose rate limits. Rate-Limiting is a softer defense against DDoS that can be deployed closer to the destination of the traffic. Currently, there are two mechanisms that employ the idea of rate-limiting on DNS servers.

[1]http://iptraf.seul.org/

1) DNS Dampening: DNS dampening obtains query data and parameters from all the requests made and assigns them penalty points. Queries most used to perform attacks or that have a larger sized response receive higher penalty points, and they will increase if the query ID keeps repeating. The server has a limit of penalty points that can be reached. If the penalty points are higher than that limit the server will start dampening, dropping all the queries coming from that IP address. The penalty points will decrease in time, and when they get to a value under the server limit the queries will start being processed again. The problem with DNS dampening is that its impossible to control the dampening of legitimate queries if they happen to reach the penalty limit. There is no mechanism to reduce the impact of false positive occurrences.

2) DNS Response Rate Limiting: Response Rate Limiting (RRL) was introduced in 2012 by Paul Vixie and Vernon Schryver [17] after observing attacks that abused DNSSEC-signed domains. The idea behind RRL is that authoritative name servers would rate limit identical outgoing responses to the same address or subnet. When the rate limit is exceeded, the server either drops the response or sets the TC (truncated) flag to signal the clients to retry using TCP instead of UDP thus denying the chance for traffic reflection.

RRL is strongly recommended for authoritative name servers which do not and should not offer recursion. Authoritative name servers are mostly queried by other name servers, usually resolvers, meaning the vast majority of their clients will properly cache the records making reasonably safe to assume that authoritative name servers would not need to repeat the same responses to same clients or subnets. Resolvers, on the other hand, are usually queried by stub-resolvers that perform very little caching of records or applications that make the requests using third-party libraries that may not perform caching at all. Because of that, resolvers may legitimately need to send identical responses to the same client.

RRL can significantly reduce the impact of DNS Amplification Attacks that abuse authoritative name servers but it also has a few limitations or disadvantages. First, attackers are able to avoid triggering rate limiting by crafting their attacks to use a large spread of queries instead of just a few very repetitive ones. Second, it can potentially affect legitimate clients that use resolvers that do not properly cache RRs. Third and most significant one, it can become itself the weapon of a DoS attack if an authoritative name server using it is flooded with queries using the source address of legitimate resolvers, for instance, the resolver of a large ISP. Clients of a rate limited resolver might see service degradation when trying to resolve names from the domain being flooded by the attackers.

IV. Response Rate Limiting

Conceptually, RRL uses a credit or token bucket scheme. To each client-response combination is assigned an account that earns credit every second that is spent every time an identical response is sent. While an account credit is below zero, responses become rate-limited and can either be dropped, truncated or sent(leaked), depending on configuration parameters. Truncating or leaking responses gives the victims a chance of still getting answers to their legitimate requests.

A. Response generation process

When generating a response, a server imputes a name to a client/subnet which can either be a wildcard name, the query name(QNAME) or the zone name.

When the server has records that match the query name it imputes the both query name and the requested RR type. If the name exists in the server but the requested RR type for it does not then only he query name is imputed and a NODATA message is sent.

When the server does not have matching records to the query name it imputes the zone name. As an example, considering two separate queries asking for the A records of ”pc1.example.com” and ”pc2.example.com”, both received by the authoritative name server of the com zone. Even though the requests are different, the answer to both is a referral message with the authoritative servers of ”example.com”. In this case, the imputed name of both responses is the zone name ”example.com”. The same behaviour applies to NXDOMAIN answers, all of them being relative to the zone name instead of the query name itself.

To keep track of clients and responses, RRL stores information in state-blobs. These are essentially key-value pairs in a table. The key is a tuple of subnet, imputed name and an error status boolean. The value is a counter for how many times the same client-response combination has been considered.

RRL goes through the following steps:

1) The sender address is masked with a subnet prefix. By default, IPv4 addresses on the same /24 subnet are grouped, while IPv6 addresses are grouped from the same /56 subnet;
2) It imputes a name and a boolean error indicator (true if RCODE is set to REFUSED, FORMERR or SERVFAIL);
3) Using the tuple <mask(ip), impute(name), errorstatus>, it looks up for a state blob, creates one if it doesn't exist and increments its counter;
4) Checks if the rate limit for that state blob has been exceeded, sending the response if it has not;
5) If the limit is exceeded, it decides dropping, truncating or leaking the response.

Figure 4 illustrates the response generation process with response rate limiting enabled.

B. Configuration parameters

The behaviour of a responder with RRL enabled can be customized through configuration parameters. Different servers may have face different scenarios, whether in the number of hosted zones, types of records it holds or overall request traffic. The RRL draft introduced the parameters that may be used to control the rate limiting behaviour [17]. They are described as follows:

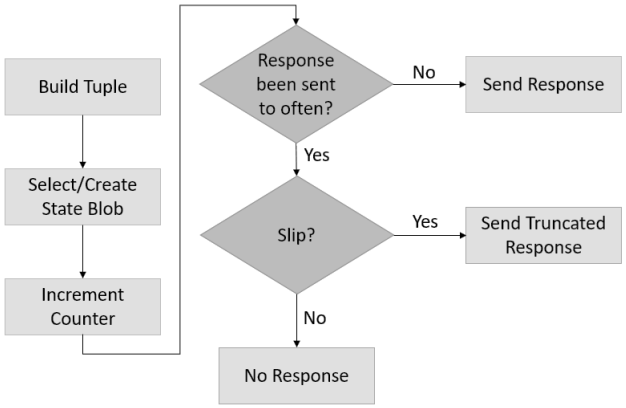

Fig. 4: Diagram of RRL response generation process.

- **RESPONSES-PER-SECOND**: the maximum allowed identical responses. A value of zero disables rate limiting for a view or restricts throttling only to NXDOMAIN and/or ERROR responses in case they are set.
- **NXDOMAINS-PER-SECOND**: sets the limit for NXDOMAIN responses. Defaults to the same as RESPONSES-PER-SECOND.
- **ERRORS-PER-SECOND**: sets the limit of error (REFUSED, FORMERR or SERVFAIL) responses.
- **WINDOW** - The period in seconds over which rates are measured. Assuming a responses-per-second value of 5 and window value of 5 then a single client is limited to 5 identical responses in any second and no more than 25 (5 x 5) identical responses within any 5 second window period.
- **IPv4-PREFIX-LENGTH**: The prefix length used to mask and group IPv4 addresses in the same bucket. Default is a /24 prefix.
- **IPv6-PREFIX-LENGTH**: The prefix length used to mask and group IPv6 addresses in the same bucket. Default is a /56 prefix.
- **LEAK-RATE**: Rate at which responses are sent normally when the client is rate limited. A value of 3 means a leaked response will be sent for every 3 dropped while the rate limit is exceeded.
- **TC-RATE or SLIP**: The same as LEAK-RATE but for truncated responses.
- **MAX-TABLE-SIZE**: Maximum number of state blobs the server can keep. It is recommended to be set to the product of WINDOW and the maximum queries per second to handle the worst case scenario where every single response is unique and requires a state blob for itself.
- **MIN-TABLE-SIZE**: The initial size to be allocated for an empty state blob table.

Support for this list is not mandatory. Depending on the DNS name server implementation, some parameters may not be available and others may be added. For instance, the most widely used name server, BIND, does not provide LEAK-RATE [18] and therefore only having support for sending truncated responses when applying rate limiting. BIND also adds ALL-PER-SECOND which sets the maximum number of all combined responses in a second, including truncated. PowerDNS and Microsoft's DNS server do not have NXDOMAINS-PER-SECOND with the latter adding MAXIMUM-PER-WINDOW which is similar to BIND's ALL-PER-SECOND but for total responses per WINDOW seconds.

V. Study Methodology

The goal of this work is to study the usage of the RRL feature in the DNS service provided by DNS.pt[2], the manager of the .pt TLD, as a mitigation mechanism against DNS Amplification Attacks that abuse but don't necessarily target its servers. The first task is to verify the behaviour of RRL when used in a server that hosts a very delegation centric domain as, for example, a TLD server. Then, it measures how effective RRL can be in reducing the impact of DNS amplification attacks.

A. Testing the effectiveness of RRL

To test the RRL feature, an isolated server is setup to host a generated zone similar to a TLD that is then hit with request generated at a client host.

The server is setup in a virtual machine running on top a Macbook Air with a dual-core hyperthreaded Intel processor. The virtual machine has 2vCPUs and 1 GB of RAM. The guest OS is Debian 9.4.0 and the name server software is BIND 9.10.3-P4. Recursion is disabled for all clients and a single zone is created on the server. Inbound and outbound network traffic on the server are measured and sampled using the IO graphing ability of Wireshark, a packet capture and network protocol analyzer.

A TLD name server usually has just one very large zone file containing the collection of all its sub-zones. For instance, the .pt zone has over 1 million[3] registered domain names in total. In an attempt to increase the sophistication of an attack, the attacker may previously gather information about the zone using webcrawlers or by conducting a zone walk[4]. To study the effectiveness of RRL on a TLD like *.pt*, a single zone file is generated containing 1000 domain names with their delegation (NS) and glue (A and AAAA) records. The zone is then signed so that the increasingly present effect of DNSSEC can also be considered.

The client host uses Python and Scapy[5] scripts to generate and send forged DNS requests at different rates. Only a single client is used since this study looks mostly at the behavior of the server making the reflection part of the attacks irrelevant. Attacks are separated into two different types: repeating query attacks in which the same request is repeated and varying query attacks in which the set of requests uses a particular

[2]https://www.dns.pt/pt/
[3]https://www.dns.pt/pt/estatisticas/?tipo=0&ordem=2&graph=0&subm=Filtrar
[4]https://www.farsightsecurity.com/2017/09/01/stsauver-zone-walking/
[5]https://scapy.net/

percentage of the existing sub-domain records, varying from 0% to a 100% of resolvable domain names.

B. Finding the ideal settings

Although there are recommended values for each setting, there is no global set of options that suites every scenario. Some of the options can be fine-tuned while using the LOG-ONLY option. In this mode, RRL does not actually drop any responses but instead logs its activity which enables post-analysis to be performed to see the effects of different settings without the risk of causing an impact on normal operation of the DNS service.

1) ***Responses per-second****:* To determine what PER-SECOND settings to use, the best approach is to analyze DNS traffic captures or logs, under normal conditions, of the specific environment. For a fully delegation-centric server, such as a TLD authoritative server, responses can be controlled with REFERRALS-PER-SECOND and NXDOMAINS-PER-SECONDS separately.

A simple solution is to determine what is the largest response that can be sent, as an example, to /24 IPv4 or /56 IPv6 subnets. With that information decide how much DNS outgoing traffic is considered acceptable for a single client or subnet. For instance, considering a maximum of 1 Mbps of DNS traffic and with an observed maximum response size of 3500 bytes then the maximum number of identical responses for that subnet can be calculated using the formula:

$$\frac{Max\ bandwidth}{Largest\ response\ size} \approx Max\ responses \qquad (1)$$

2) ***Window size****:* The WINDOW setting acts as the punishment for an offending client. A bigger punishment means the client can start receiving normal answers after a longer period. If RESPONSES-PER-SECOND has a low setting and WINDOW is high, the DNS server will be extremely protective, assigning long punishments to rather low offenses. The opposite is a DNS server that is slow to start rate limiting offenders and fast at letting them receive answers. Preferably, it should be kept to smaller values meaning victims of attacks can return to normal behavior quicker.

3) ***IPv4 and IPv6 prefix length****:* RRL groups clients into buckets because it is protecting them at a distance. The default values of /24 for IPv4 and /56 IPv6 represent small enough groups that their DNS traffic can be combined into a single bucket while avoiding false positives. Smaller length prefixes will group more clients together increasing the chance of legitimate clients being grouped with offenders. Bigger length prefixes do the opposite, decreasing that chance.

4) ***SLIP****:* With the SLIP setting there is a trade-off between how much outgoing traffic to allow and chances of false positives. Higher values for SLIP will result in less outgoing traffic but increase the chance of a legitimate request being dropped, i.e., a false positive occurrence. The chance is related to the SLIP setting in the following expression:

TABLE I: Probability of false positives for different SLIP values

SLIP	False positive probability	TCP response probability
SLIP 1	0%	100%
SLIP 2	50%	≈ 97%
SLIP 3	66.6%	≈ 87%
SLIP 5	80%	≈ 67%
SLIP 10	90%	≈ 41%

$$\frac{SLIP-1}{SLIP} \times 100 = Chance\ of\ false\ positive \qquad (2)$$

An increased chance of having legitimate requests being dropped also means a lower chance for clients to retry using TCP. Setting SLIP to a value higher than 1 means less responses with the TC bit set are sent, decreasing the chance for the client to receive one. The chance that a rate-limited client reconnects and receives a TCP response can be seen as a binomial probability of having 1 successful attempt out of a maximum of n retries with the probability of success being equal to chance of receiving a response with the TC bit set.

Table I shows the probability of false positive occurrences for different values of SLIP and the corresponding probability of a client receiving a TCP response assuming the client does 5 UDP retries before giving up.

Due to security concerns regarding DNS cache poising at resolvers [19], the recommended value for SLIP is 1. With a SLIP value of 1 there are no false positives. Responses are still rate-limited, meaning amplification is still cut but for every request, a truncated response is sent while an attack is occurring. SLIP 1 completely eliminates the chance of false positive ocurrences at the cost of reflecting all the traffic. If a repeated request is being sent at a rate of 100 Mbps, a DNS server configured with a SLIP value of 1 will send approximately 100 Mbps worth of truncated responses to the victim.

VI. Measurements and Results

A. RRL behaviour in a TLD-like zone

A TLD is a special type of zone. Its zone file should only contain the delegation records for the domain names registered under the TLD. This means the server holds only NS records and, in most cases, their respective glue A and AAAA records. This means that all of the requests for resolvable domains names should return referral responses. To verify this behaviour RRL was enabled on the test server with default values and the log-only option enabled. The client host sends requests using random QNAMEs under the same domain-name at a rate of 10 requests per second which is enough to trigger rate-limiting for the demonstration.

Listing 1: Grouping of different qnames for the same referral response.

```
root@ns1:~# tail -n 5 /var/log/named/queries.log
```

```
... (hclgtixc.67.tld): query: hclgtixc.67.tld IN A +
... (piruzcoa.67.tld): query: piruzcoa.67.tld IN A +
... (ndhpquof.67.tld): query: ndhpquof.67.tld IN A +
... (klmgudbx.67.tld): query: klmgudbx.67.tld IN A +
... (stfvcqle.67.tld): query: stfvcqle.67.tld IN A +
root@ns1:~# tail -n 5 /var/log/named/rate-limit.log
... (hclgtixc.67.tld): would rate limit drop referral ... for 67.tld IN
... (piruzcoa.67.tld): would rate limit slip referral ... for 67.tld IN
... (ndhpquof.67.tld): would rate limit drop referral ... for 67.tld IN
... (klmgudbx.67.tld): would rate limit slip referral ... for 67.tld IN
... (stfvcqle.67.tld): would rate limit drop referral ... for 67.tld IN
```

In the log lines on listing 1 we can see different, non-repeating requests for existing records being received by the server. The behaviour of a TLD server is observed here as every request, although different, results in the same response which is a referral for "67.tld" with information for its authoritative name servers. This shows that RRL in a TLD zone will group responses by domain name regardless of what record type is being requested since all the server will ever do is send referral responses.

A similar behaviour is observed when the requests are made for non-existing domains on the TLD. The sample on listing 2 shows completely different requests being received by the server resulting in NXDOMAIN responses. Because NXDOMAIN responses are not bound to a domain name or set of domain names, they are seen and grouped under the TLD name meaning that all NXDOMAIN responses to a single client or subnet are treated as the same.

Listing 2: Grouping of NXDOMAIN responses for the whole TLD.

```
root@ns1:~# tail -n 10 /var/log/named/queries.log
... (ephpsffzgb.205851.tld): query: ephpsffzgb.205851.tld IN A +
... (sorzqloepv.256681.tld): query: sorzqloepv.256681.tld IN NS +
... (niqvmmygdo.239178.tld): query: niqvmmygdo.239178.tld IN NS +
... (sguupwjsnr.294212.tld): query: sguupwjsnr.294212.tld IN AAAA +
... (oxofpeeskj.256061.tld): query: oxofpeeskj.256061.tld IN AAAA +
root@ns1:~# tail -n 10 /var/log/named/rate-limit.log
... (ephpsffzgb.205851.tld): would rate limit drop NXDOMAIN ... for tld
... (sorzqloepv.256681.tld): would rate limit slip NXDOMAIN ... for tld
... (niqvmmygdo.239178.tld): would rate limit drop NXDOMAIN ... for tld
... (sguupwjsnr.294212.tld): would rate limit slip NXDOMAIN ... for tld
... (oxofpeeskj.256061.tld): would rate limit drop NXDOMAIN ... for tld
```

B. Repeating query attacks

The most simple attacks to craft are the ones that repeat the same request over and over. These can be referred as repeating query attacks and it will usually try to maximize amplification by making requests that are known to provide a great deal of amplification. The ANY request is the most commonly abused RTYPE in DNS Amplification attacks [3] since it triggers the server to return all resource records it has for the domain name.

Figure 5 shows a measurement of inbound and outbound traffic on the test server. A repeating ANY query attacks begins flooding the server resulting in an average of 600 Kb/s of incoming traffic and 3.5 Mb/s of outgoing traffic. RRL is enabled and starts rate-limiting to 10 responses per second with a SLIP setting of 1 which is then increased to 2. Table II shows the effect of RRL using different values for SLIP against a repeating query attack. As expected, when RRL is enabled with SLIP=1 it reduces outgoing attack traffic to the same level of incoming traffic as the repeated responses are now truncated and therefore the same size as the corresponding requests. When SLIP is set to 2, outgoing offending traffic is reduced even more due to only 1 out of every 2 responses being sent truncated.

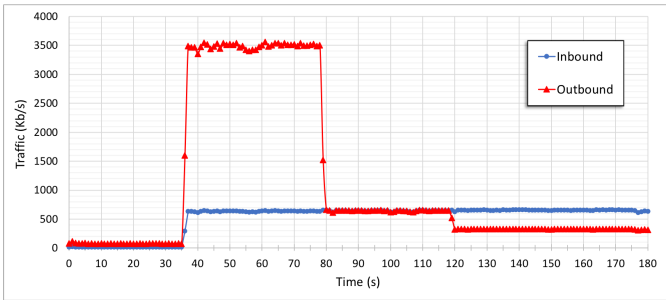

Fig. 5: A repeating query attack against a test server hosting a fictitious zone similar to a TLD. RRL is enabled at 80 seconds using SLIP=1 and then SLIP=2 at 120 seconds.

TABLE II: Outgoing traffic volume of repeating query attacks.

SLIP	False positive probability	Incoming traffic	Outgoing traffic
SLIP 1	0%	618 Kb/s	625 Kb/s
SLIP 2	50%	625 Kb/s	312 Kb/s
SLIP 3	66.6%	630 Kb/s	210 Kb/s
SLIP 5	80%	635 Kb/s	126 Kb/s
SLIP 10	90%	640 Kb/s	64 Kb/s

RRL effectively stops the not so sophisticated attacks, that rely almost entirely on a single query to maximize amplification. This is the ideal scenario for RRL effectiveness and the results on table II show that as outgoing traffic is reduced to almost exactly what is expected for each SLIP value.

C. Varying query attacks

Repeating the same request over and over is a simple flooding attack. RRL was designed from the premise that authoritative name server should not have to repeat the same answer frequently. In this section three experiments are conducted to understand how effective RRL can be when the sophistication of attacks increases. By randomly changing the requested record name and type, an attack may avoid triggering rate-limiting for its intended victim. The request records can either be randomly generated names or randomly picked from already known existing records. Prior to an attack, the attacker can gather information about the a zone by abusing NSEC records and performing a so called zone walk.

1) NXDOMAIN abuse (0% resolvable domain names): The first experiment consists of send varying requests that result in NXDOMAIN responses. Although the requests and responses are different, RRL groups all NXDOMAIN responses for a single zone on the same "bucket" (see VI-A). Figure 6 and table III shows RRL reducing outbound traffic from an attack by which all requests result in NXDOMAIN. The results resemble the obtained ones with the repeating query attack, since outgoing NXDOMAINs are all treated as the same response, therefore triggering rate-limiting for all NXDOMAIN responses for that client.

2) Mixed attack (50% resolvable domain names): In the second experiment, a slightly more sophisticated attack results in 50% NXDOMAIN responses and 50% unique NOERROR

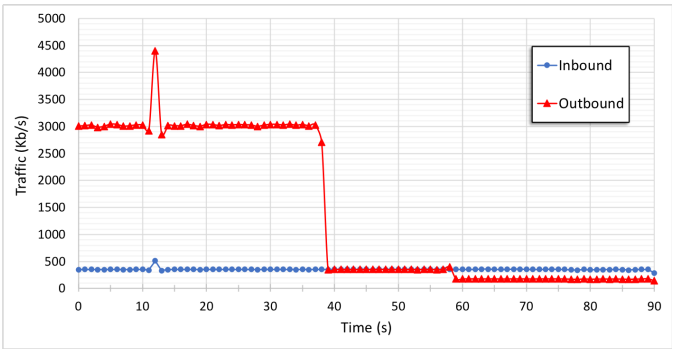

Fig. 6: A varying query attack that results in 100% NXDOMAIN responses. RRL is enabled at 38 seconds using SLIP=1 and then SLIP=2 at 58 seconds.

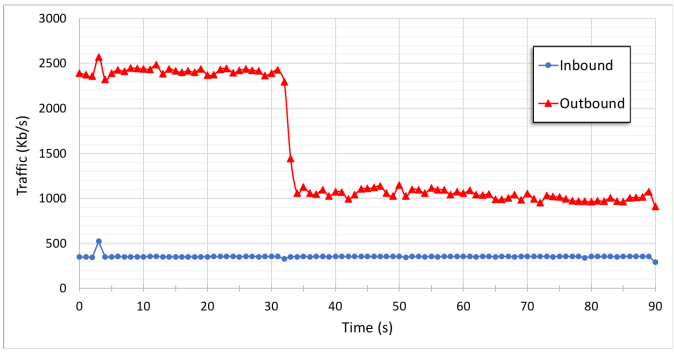

Fig. 7: A varying query attack that returns a mix of NXDOMAIN and NOERROR responses. RRL is enabled at 30 seconds using SLIP=1 and then SLIP=2 at 60 seconds.

TABLE III: Outgoing traffic volume of an attack that results in NXDOMAIN responses.

SLIP	False positive probability	Incoming traffic	Outgoing traffic
SLIP 1	0%	347 Kb/s	347 Kb/s
SLIP 2	50%	347 Kb/s	174 Kb/s
SLIP 3	66.6%	348 Kb/s	115 Kb/s
SLIP 5	80%	347 Kb/s	69 Kb/s
SLIP 10	90%	348 Kb/s	35 Kb/s

responses for existing records. This covers the scenario of an attacker having some knowledge of records it can abuse on the zone. In order to avoid triggering rate-limit for the known records requests, the attacker mixes the attack with generated names that result in NXDOMAIN responses. Figure 7 and table IV show that having knowledge and using a larger set of different requests results in higher outgoing traffic. Roughly 50% of the responses avoid getting rate-limited as they are not repeated very often. The other 50% that result in NXDOMAIN are still grouped but because they are now less frequent, triggering rate-limit on NXDOMAIN also does not happen as easily. Taking into account the generated domain is signed, NXDOMAIN responses are much more amplified than requesting, for example, existing A records. NXDOMAIN responses with DNSSEC carry, in most cases, 3 NSEC3 records to prove denial of existence. If a percentage of these responses can escape RRL while being dilluted with other requests it also adds to the overall amplification ratio of the attack.

TABLE IV: Outgoing traffic volume of a mixed query attack.

SLIP	False positive probability	Incoming traffic	Outgoing traffic
SLIP 1	0%	347 Kb/s	1 Mb/s
SLIP 2	50%	348 Kb/s	940 Kb/s
SLIP 3	66.6%	347 Kb/s	920 Kb/s
SLIP 5	80%	347 Kb/s	892 Kb/s
SLIP 10	90%	347 Kb/s	881 Kb/s

3) Fully-indexed zone abuse (100% resolvable domain names): The worst case scenario may happen when the attacker has managed to index all of the records in the zone. Figure 8 and table V demonstrate that an attack using a very large set of unique requests may completely avoid being rate-limited. This raises a question of how RRL effectiveness may decrease when the number of records in a target zone or name server increases. The more information an attacker acquires about a sufficiently large zone(millions of records) the more easily its attack can fly under the radar of RRL.

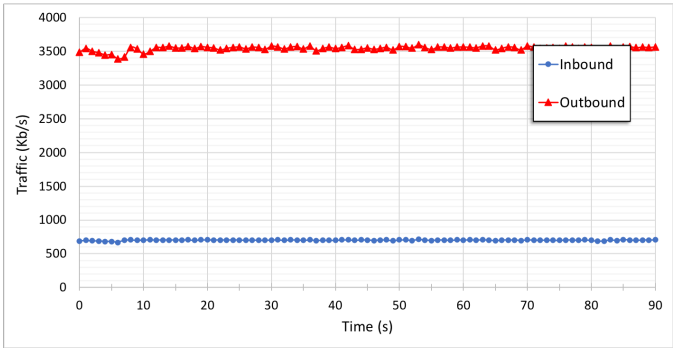

Fig. 8: A varying query attack that uses all the records of the zone. RRL is enabled at 30 seconds using SLIP=1 and then SLIP=2 at 60 seconds.

TABLE V: Outgoing traffic volume of attack randomly requests one of all the zone records.

SLIP	False positive probability	Incoming traffic	Outgoing traffic
SLIP 1	0%	630 Kb/s	2.45 Mb/s
SLIP 2	50%	630 Kb/s	2.4 Mb/s
SLIP 3	66.6%	630 Kb/s	2.38 Mb/s
SLIP 5	80%	631 Kb/s	2.36 Mb/s
SLIP 10	90%	630 Kb/s	2.34 Mb/s

VII. Conclusion

DNS amplification attacks rely on two points, reflection and amplification. The DNS protocol is mostly used over UDP meaning that the source IP address can be easily spoofed, making DNS servers traffic reflectors. Additionally, the DNS

protocol has an amplification behaviour in its request-response mechanism. Small requests are able to be returned as very large responses. The adoption of DNSSEC greatly contributes to this amplification effect since it adds very large records, necessary to authenticate information on the client side. By controlling a large number of bots, attackers can achieve network traffic volumes in excess of 1 Tbps [1].

To tackle the issue, RRL was proposed and recommended for usage on authoritative DNS servers. RRL limits the number of similar responses within a configurable window. If the same response is seen too many times for the same client within the window period, RRL triggers rate-limiting during a certain time, stopping potential offending traffic from coming out of the DNS server. In this work, in a lab setting, the RRL feature has shown to be effective against common, yet simple attacks that repeat the same request in an attempt for the greatest amplification ratio possible on the server. However, its effectiveness starts decreasing as attacks begin to use larger sets of requests. If an attacker knows a large enough set of existing domais names within a zone, it may be able to completely avoid triggering rate-limiting. Using the SLIP=1 setting, false positives can be avoided by never dropping responses but simply truncate them. Increasing the SLIP value may reduce outbound traffic but at the cost of denying service to legitimate clients as they may not be able to reconnect using TCP.

Even though it does not pose as a great barrier against smart and sophisticated attacks, RRL does have a positive effect on authoritative name servers with minimal side effects for either legitimate clients or the servers. When configured to never drop responses, there are no noticeable disadvantages comparing to not having it enabled. In the best scenario it can completely stop the common repeating query attack and in the worst scenario it acts as if it was not enabled.

VIII. Future Work

One particular challenge of this work was the fact that it was not carried on real production DNS servers so it could not study how a baseline of traffic from a real TLD server actually affects RRL and vice-versa. In other words, it did not evaluate RRL in a highly heterogeneous environment of clients/resolvers and traffic patterns. In such environment, it is difficult to establish a baseline or traffic pattern per client and response which is what the per-second settings of RRL should portray. An addition that RRL could benefit from is a process in which continuous analysis of traffic patterns causes the mechanism to automatically adjust its per-second limits to either allow abnormal yet legitimate client behaviour or stop large volume attacks that currently avoid triggering rate-limiting.

References

[1] S. Khandelwal. (2016, September) World's largest 1 tbps ddos attack launched from 152,000 hacked smart devices. [Online]. Available: https://thehackernews.com/2016/09/ddos-attack-iot.html

[2] D. Goodin. (2016, September) Record-breaking ddos reportedly delivered by over 145k hacked cameras. [Online]. Available: https://arstechnica.com/security/2016/09/botnet-of-145k-cameras-reportedly-deliver-internets-biggest-ddos-ever/

[3] US-CERT. (2013) Ta13-088a : Dns amplification attacks. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA13-088A

[4] "Domain names - implementation and specification," RFC 1035, Nov. 1987. [Online]. Available: https://rfc-editor.org/rfc/rfc1035.txt

[5] D. C. MacFarland, C. A. Shue, and A. J. Kalafut, "Characterizing optimal dns amplification attacks and effective mitigation," in *International Conference on Passive and Active Network Measurement*. Springer, 2015, pp. 15–27.

[6] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse." in *NDSS*, 2014.

[7] A. J. Kalafut, C. A. Shue, and M. Gupta, "Touring dns open houses for trends and configurations," *IEEE/ACM Trans. Netw.*

[8] R. Rosenbaum, "Using the Domain Name System To Store Arbitrary String Attributes," RFC 1464, May 1993. [Online]. Available: https://rfc-editor.org/rfc/rfc1464.txt

[9] "Security Bulletin: Crafted DNS Text Attack," Akamai's Prolexic Security Engineering and Research Team (PLXsert), Tech. Rep., 11 2014.

[10] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "Dnssec and its potential for ddos attacks: a comprehensive measurement study," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 449–460.

[11] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827 (Best Current Practice), Internet Engineering Task Force, May 2000, updated by RFC 3704. [Online]. Available: http://www.ietf.org/rfc/rfc2827.txt

[12] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets," in *ACM SIGCOMM computer communication review*, vol. 31, no. 4. ACM, 2001, pp. 15–26.

[13] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, p. 3, 2007.

[14] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 312–321.

[15] ——, "Source-end ddos defense," in *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on*. IEEE, 2003, pp. 171–178.

[16] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "Detecting dns amplification attacks," in *International Workshop on Critical Information Infrastructures Security*. Springer, 2007, pp. 185–196.

[17] P. Vixie and V. Schryver, "Dns response rate limiting (dns rrl)," *URL: http://ss. vix. su/˜ vixie/isc-tn-2012-1. txt*, 2012.

[18] ISC. (2017) Bind 9 administrator reference manual. [Online]. Available: https://ftp.isc.org/isc/bind9/9.12.0a1/doc/arm/Bv9ARM.pdf

[19] Florian Maury, Mathieu Feuillet. Blocking dns messages is dangerous. ANSSI. [Online]. Available: https://www.ssi.gouv.fr/uploads/IMG/pdf/DNS-OARC-2013-Blocking$_D NS_M essages_I s_D angerous.pdf$