Estimating the Amplification Factor of Three Common Protocols in DRDoS Attacks

A Quantitative Analysis on the Weaponisation of Hosts Located in Greece

Rares-Andrei Toader[1]

Supervisors: Georgios Smaragdakis[1]**, Harm Griffioen**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Rares-Andrei Toader
Final project course: CSE3000 Research Project
Thesis committee: Georgios Smaragdakis, Harm Griffioen, Georgios Iosifidis

Abstract

Distributed reflection denial-of-service (DRDoS) attacks are a type of cyberattack where a malicious actor sends requests to public and open servers on behalf of the victim by spoofing their IP address. The traffic generated by the corresponding responses is directed towards the victim (whose IP address appeared as the source address of the initial packets), potentially exhausting their bandwidth. These attacks have kept becoming more powerful over the years.

This thesis presents a measurement study of three well-known protocols, where we assess the amplification potential of hosts located in Greece running these protocols. We find that DNS remains the most vulnerable protocol to amplification; the top 250 hosts can cumulatively amplify the traffic by 32,000×. Furthermore, we discover that the "ANY" query type and the improperly configured DNS extension (EDNS0) are two significant causes of DNS amplification. Lastly, we also find hosts vulnerable to looping attacks, a novel threat in the context of DDoS attacks.

I. Introduction

A *denial-of-service attack* (DoS) is a collection of cyber attacks that aim to disrupt the availability of a system [1]. This attack and its variants have been widely researched [2] in the literature and have historically been one of the most popular strategies used by adversaries [3].

A *distributed denial-of-service* (DDoS) attack has the same goal as a normal DoS attack [4]. However, it achieves this by having multiple adversarial entities exhausting the target's resources instead of a single attacking entity. DDoS is a more powerful attack than the normal DoS [4]. The attacking entities in a DDoS context could be compromised computers or infected IoT devices, which could be part of a larger, malicious entity called a *botnet* [5]. The target can be a web server or another network service. The flood of incoming packets or connection requests to the victim's system forces it to become slower, become unresponsive or even crash, denying regular service to legitimate users, thus violating the "A" (availability) from the CIA (Confidentiality, Integrity and Availability) triad [6].

In a *distributed reflection* DoS attack (DRDoS), a malicious actor uses public services to reflect traffic to a victim [7]. The attacker sends requests on behalf of the victim, and in turn, the victim gets flooded with responses from the public services. Consequently, if enough requests are made and the size of the responses is large, the target's bandwidth is depleted. An amplification attack is essentially a DRDoS attack, where the attacker focuses on sending requesting packets to the public hosts, whose size would be smaller than the responding packets sent to the victim. In this sense, the response packets amplify the requesting packets, hence the attack's name.

Several reasons make amplification attacks possible. IP spoofing in UDP-based protocols makes it possible for an attacker to send packets on the victim's behalf. *IP spoofing* means the attacker sets the source address in the IP header as the victim's IP address when sending a packet to a public server. Consequently, the victim receives the response packets from the servers. This happens because UDP-based protocols respond to queries without being required to establish a connection (which is the case for TCP-based protocols) and thus do not validate the IP address of the entity sending the request. IP spoofing is considered a bad practice, and measures for it have been described 24 years ago in an RFC (Request for Comments) [8]. This introduces the idea of network ingress filtering (commonly referred to as BCP38) [8], which would be a general measure against denial-of-service attacks that utilise IP source address spoofing. However, empirical evidence shows that IP spoofing is an ongoing problem that has yet to be eliminated [9], remaining the most critical cause of amplification attacks [7].

Another essential cause for amplification attacks is an asymmetry between the size of a request to a public server and the size of the response. This asymmetry enables amplification and is a desirable feature for an attacker, as they only have to send a small packet to reflect a larger amount to the victim. This security issue is present in many protocols because these protocols have not been implemented with this aspect in mind. For example, the "monlist" command [10] allows users to query an NTP server, retrieving up to 600 hosts with which the NTP server has most recently communicated. This feature was mainly meant as a debugging tool but has also been weaponised by adversaries in successful DRDoS attacks [7].

The first DDoS attack happened in July 1999, more than 20 years ago [11]. The biggest DDoS attack ever recorded took place in 2017 [12]. It targeted Google services and peaked at 2.54 Tbps. DDoS attacks have kept getting more notoriety over the years, becoming well-established cyber attacks with a popularity that has kept rising. In a recent DDoS threat report for 2024 Q1, Cloudflare mentions that they have noticed a 50% increase in DDoS attacks compared to the corresponding first quarter in 2023 [13]. DDoS attacks are extremely popular in the cyber threat landscape for several reasons. Firstly, they are relatively easy to orchestrate, requiring little technical expertise, as they can even be bought online as a service, often under the umbrella name of "stress-testing", at a reasonable cost [14]. Secondly, these attacks effectively overwhelm a victim's system, leading to downtime or service disruption. Moreover, the malicious actor can achieve this while also preserving their anonymity.

Seeing the impact and complexity of DRDoS attacks, we realised there is no auditing tool to find servers in an autonomous system (AS) [15] that could be weaponised and, consequently, unintentionally participate in amplification attacks. The literature lacks a comprehensive list that enumerates all the negative aspects of all existing protocols or the configuration of a specific server that could contribute to launching an amplification attack.

As an extension, we also analyse looping attacks, a novel DoS attack at the application-layer level [16]. This particular attack leads to infinite amplification, as two vulnerable servers would send messages to each other indefinitely. This happens, for instance, when a host responds to a malformed request with an error message, and this response prompts another server to send the packet needed for the initial server to send the same error message. Thus, a traffic loop is formed between the two servers. This type of attack can be hazardous, as it could also be used to target network links. We follow the methodology in [16] to discover which servers are vulnerable and how many vulnerable host pairs could be formed. Note that our study focused on Greece, as this was the country assigned to us for our project. With this thesis, we make the following contributions:

- We investigate which DNS, NTP, and Memcached servers located in Greece can be utilised in amplification attacks.
- We explore what amplification factor can typically be achieved with the Greek infrastructure explored and what conditions influence this factor.
- We measure the number of vulnerable pairs of DNS and NTP servers located in Greece that could be formed in the context of looping attacks.

The thesis is structured as follows. In Section II, background information is given, such as definitions and important concepts. Section III mentions notable contributions related to our work. Section IV describes the data used in this work and how it was obtained. Section V presents the ways in which the experiments and measurements were performed. Section VI mentions the operational considerations enforced throughout this work in order to follow an ethical research procedure. Section VII highlights the results, and Section VIII showcases reflections on them. Lastly, Section VIII draws the conclusions.

II. Background

This section provides details and definitions of relevant concepts in this thesis. We also give definitions and ways the considered protocols were used in amplification attacks in the past.

An *amplifier* is a public server running a UDP-based protocol. With this consideration, the server can receive spoofed requests and reflect the response to the victim. This paper considers servers running one of the following three services: the Domain Name System (DNS), the Network Time Protocol (NTP) and Memcached. The motivation behind restricting the research to these three services is that they have been notorious in the context of DRDoS, being potent amplification vectors [17], [18], [19]. Nonetheless, for further work, other protocols are worth exploring, as shown in [7], such as the Simple Network Management Protocol (SNMP) [20].

To measure how powerful an amplifier is, we use the *bandwidth amplification factor* (BAF) as this has been previously defined in the literature in the context of amplification attacks [7]. BAF is defined as the quotient between the size of the UDP payload of the amplifier answer and the size of the UDP payload of the spoofed request, see Equation (1).

For gathering information about hosts, such as IP addresses and services running on that server, we have used *Censys* [21]. It is a search engine that enables security researchers to find and analyse devices and networks on the Internet quickly, without a heavy computational effort (via an academic license that we also received). Censys gathers data on hosts through periodic Internet scans, the scanning method relying on a variant of ZMap. This is a revolutionary scanner that can scan the entire IPv4 address range in around 45 minutes with enough bandwidth resources [22]. Censys scans and stores the ports of servers that are publicly accessible. The platform allows queries such as retrieving all hosts in Greece or hosts with a service running on a specific port. Queries used for this thesis along with some extra information are shown in Appendix A.

Protocol Overview. The *Domain Name System* (DNS) is often described as the "phonebook" of the Internet [23]. DNS forms the critical infrastructure that helps the browser find the assigned IP address of the server where the website is hosted, given the domain. DNS enables users to use simple, human-readable domain names that can be easily remembered and translated into IP addresses. This translation process is the so-called *address resolution*. DNS usually runs on port 53 and can

$$BAF = \frac{len(UDP\ payload)\ amplifier\ to\ victim}{len(UDP\ payload)\ attacker\ to\ amplifier} \quad (1)$$

handle many other types of requests, responding with corresponding resource records (RRs). If someone sends an "ANY" request, the DNS server answers with all the resource records related to the domain queried. For this reason, DNS represents an attractive amplification vector for malicious adversaries. It does not surprise that according to Cloudflare, in 2021, 44% of organisations mentioned that attacks based on DNS were one of their most challenging security problems [24].

The *Network Time Protocol* (NTP) is a protocol used to synchronise the clocks between computers over a network [25]. Essentially, it ensures that all the computers connected to a network agree on the time, up to an accuracy of milliseconds. This is crucial for tasks that require precise timing, such as logging events or scheduling tasks. Implementations of this protocol commonly use the port number 123 and exchange timestamps over UDP [25]. NTP has been weaponised previously via the "monlist" command, which returns a list of up to 600 hosts that most recently communicated with an NTP server [18]. This is prone to creating significant responses compared to a small query, leading to an enormous amplification factor. Mitigations have been proposed; Kührer et al. explored this vulnerability and managed to carry out a campaign that, in the end, reduced the susceptible NTP servers by a staggering 92% [26]. We aim to investigate whether the NTP "monlist" command is still a threat to the Greek network infrastructure.

Memcached is a distributed memory caching system primarily used to increase the performance of dynamic web applications by removing some load from the database [27]. It works by storing data from the result of database calls, API calls, or page rendering in memory (essentially being cached). The data is stored in the random-access memory (RAM) of the servers running the Memcached daemon and is structured as a sizeable hash table distributed across machines [28]. This means that when the same data is needed again, it can be retrieved faster from this memory cache rather than being recomputed or fetched from a slower disk-based database. Memcached services often run on port 11211 and sometimes respond to UDP requests [28]. GitHub sustained a heavy DRDoS attack in 2018 based on the Memcached "get" query. This attack was described by Akamai in 2018, and it presented how such a query could lead to a potential amplification factor of $50,000$ [29].

III. Related Work

This section describes existing research in the field of amplification attacks. Rossow [7] did a comprehensive study of how suitable 14 popular UDP-based protocols are for amplification attacks. The results of this study were quite worrying, as the author found millions of amplifiers that could be used in a distributed attack for six protocols. We similarly assess the amplification potential of DNS and NTP. The paper also explores methods to detect DRDoS attacks in the wild using traffic analysis. Lastly, Rossow also presents DRDoS mitigation techniques and ways to strengthen network protocols against this type of attack.

Griffioen et al. [30] explored the steps an attacker takes when orchestrating an amplification attack by observing a set of honeypots deployed in the wild. We followed a methodology akin to the one the authors presented. Their work also discovered that attackers have a shared memory of previously used servers in amplification attacks. Still, attackers differ in

how they orchestrate such an attack, the landscape being filled by both novice actors and very sophisticated adversaries.

Van der Toorn et al. [31] researched the amplification potential of domain names in the context of "ANY" queries, and what the reduction in response size would be if "ANY" queries would be dropped, according to the suggestions of RFC 8482 [32]. They also proposed and validated a technique to approximate the size of responses to "ANY" requests based on a large active DNS measurement dataset. We crafted our DNS query (the one we measure the amplification factor with) following what the authors also proposed.

Kührer et al. [26] studied the diversity of amplification sources, finding that they spread across various operating systems. A crucial impact of their paper was that they carried out a large-scale campaign to alert NTP administrators about the amplification potential of answering the "monlist" request. As a result, this campaign led to more than a 92% decrease in vulnerable NTP servers. Their work also highlights vulnerabilities in the TCP handshake that attackers could use as amplification vectors, which we will not explore as it is outside the scope of our research. They conclude that the root cause for amplification attacks is the fact that there are networks that allow IP address spoofing.

Pan et al. [16] researched a novel and hazardous type of denial-of-service attack at the application layer. The *looping attack* allows a malicious actor to send one IP-spoofed packet to a server that, in turn, triggers an infinite loop of back-and-forth messages between the server to whom the packet was sent and the server whose IP was spoofed in the initial request. This essentially enables infinite amplification. They propose a systematic methodology to search for vulnerable server pairs. They further use this methodology to investigate loops that can be formed with servers running various UDP protocols. The results were quite jarring, as an attacker could potentially abuse billions of server pairs for looping attacks. For this reason, we chose to study if hosts located in Greece are also susceptible to this type of attack.

IV. Datasets

This section presents the sources of the data used for our research. Four primary data sources were inspected. The first data source is Censys [21] with their Internet database, and the second is IANA [33] to find all top-level domains. The third is NTP Pool [34] and the fourth source is an archived list of the most popular domain names from the Technical University of Munich (TUM) [35].

Host information. Censys offers a large variety of publicly available information related to the public Internet. The platform allows for queries similar to those in a search engine. We used this functionality to query for hosts located in Greece (via passive scanning), which are running DNS, NTP, and Memcached on ports 53, 123 and 11211, respectively. The three queries (one per protocol) can be seen in Appendix A. The output from the queries came as raw JSON files containing a list of IP addresses alongside relevant metadata such as location or AS. After processing the data, we were left with 7,532 (DNS), 25,962 (NTP) and 13 (Memcached) servers.

The data was collected on May 10, 2024. Still, since we employed passive scanning (we just queried whatever data was stored; we did not actively probe servers at this stage), we are unaware of the actual date of origin when Censys collected the data.

Top-level domains. The *Internet Assigned Numbers Authority* (*IANA*) is an organisation that is responsible for allocating IP addresses, AS numbers and other tasks [33]. We used it to gather a list of all available top-level domains (TLDs). This data was collected on May 2, 2024.

The list of all TLDs is later ranked based on which domain is associated with the largest response size. That domain is later used in DNS "ANY" queries. The rationale behind this decision is that ranking all possible domains requires much more computational effort and is outside the scope of this work. However, this is an interesting topic, which has been researched by Van der Toorn et al. [31]. In this work, they rank hundreds of millions of domain names based on the amplification factor they can achieve. Another insightful finding they discovered is that the domains used in DNS-based DDoS attacks are among the largest ones that could be used. Furthermore, even larger domains that had yet to be exploited were found.

NTP Pool servers. The NTP Pool project represents a sizeable virtual cluster of NTP servers distributed worldwide [34]. This pool is critical in synchronising clocks for millions of machines worldwide. We found 12 NTP servers located in Greece that were part of this group as of May 3, 2024. We considered that they were worth testing for amplification since many clients use the pool. Although all 12 NTP servers were open, none of them were found to be vulnerable to amplification.

Most popular domain names. From the archive by TUM, we obtained the top 10 million domain names according to Open PageRank as of April 16, 2023 [35]. This dataset was used to find popular Greek domain names. We kept the most popular 1,023 Greek domains. Consequently, these helped us find authoritative nameservers located in Greece.

Using the first three datasets, we were able to make precise measurements of the amplification potential of DNS, NTP, and Memcached hosts located in Greece. The last dataset enabled us to see whether authoritative nameservers can serve as better amplifiers than non-authoritative ones in the context of DNS-based amplification attacks.

V. Methodology

In this section, we present the methodology for measuring the amplification potential of servers located in Greece, alongside a brief description of how we counted the number of server pairs vulnerable to looping attacks. An overview of our methodology can be seen in Fig. 1.

Measuring the amplification potential. To accurately measure the BAF of hosts located in Greece running DNS, NTP or Memcached, we have followed a similar methodology to the one presented by Griffioen et al. [30]. We mimicked an adversary's workflow without actually launching a DDoS attack.

Filtering closed hosts. Firstly, we have gathered and processed host information (the so-called "Reconnaissance" phase). An attacker would first be interested in seeing whether these servers are "online", i.e. accepting and responding to requests, as this is the prime precondition to be weaponised. This step allowed us to avoid doing unnecessary work on offline hosts. This step reduces the computational load on our machine and any strain we might put on the network. Thus, a filtering step would be essential if such a measurement study was conducted worldwide. More details for each filtering packet presented below can be seen in Appendix B.

We hand-crafted packets for each protocol, which we then sent to hosts, and filtered out servers that did not respond with

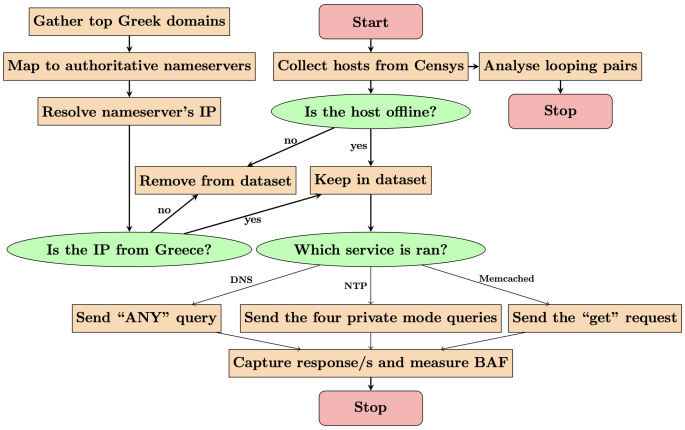

Fig. 1: Methodology diagram. The top left side corresponds to collecting authoritative nameservers. The top right part shows how to find the server pairs vulnerable to looping attacks using the methodology from the correspondent paper.

a packet we expected, for instance, returning an ICMP error packet or hosts that did not respond within a timeout period we set. The packets we crafted were aimed to be basic, common packets that an open server would be expected to answer without any issue. This measure is by no means foolproof and could be improved by sending more than one probe per protocol.

DNS. We send a simple request (query type = "A") to resolve the IP address of "google.com", with the rationale that this domain is widespread and almost any open recursive DNS resolver would be able to answer it (by retrieving the appropriate record from the corresponding authoritative nameserver). Recursive resolvers, as opposed to authoritative nameservers, do not hold the answer RRs, but they make the necessary requests on behalf of the client (if the result was not cached). In this part, we were aware that we might be biased towards filtering out authoritative NSs since they are less likely to hold the "A" RR requested (a limited number of servers have this record). They might still retrieve them if they forward the request to another resolver in case of an error. However, this is unexpected behaviour and a bad practice [36]; even an RFC from 2008 mentions that, in general, authoritative NSs should not provide recursion [37].

The bias to discard authoritative NSs is hard to measure since DNS servers do not respond, admitting they are authoritative. To mitigate this, we performed a procedure to complement our DNS host set, which will be explained later, by collecting authoritative NSs. We reject responses with "rcode" set to 2 or 5, which stands for server failure and refused, respectively. We also set a timeout of 400 milliseconds. The filtering step left us with 2,397 servers.

NTP. We create a simple Mode 3 packet, i.e. a "client mode" packet. This is one of the several operational modes of NTP, and it allows a "client" to query a server for the current time. In a sense, it represents the core functionality of NTP. To validate this approach, we looked at how the "sntp" command-line utility tool [38] crafts packets (by capturing the packets sent and inspecting them via Wireshark [39]), and we noticed they send the exact packet we crafted with Scapy, a robust packet manipulation tool with Python support [40]. We rejected any non-NTP response and set a timeout period of 200 milliseconds. This step left us with 25,130 servers. We then proceeded with another filtering stage, as we are only interested in Mode 7 queries (they lead to the highest amplification); we also sent the notorious "monlist" command by setting the request code to 42 [10]. A timeout period of 1 second was set, and any non-NTP response packets were not considered. The dataset was further reduced to 633 servers.

Memcached. A "stats slabs" packet is created to query the Memcached server for detailed statistics about the memory allocation. Memcached uses a slab allocation technique to manage memory efficiently for storing items of various sizes [41]. Slabs are essentially blocks of memory divided into chunks. Each chunk can hold a cached item. When this packet is sent, Memcached is supposed to answer with information about each slab class, including the number of chunks allocated or used. After this was carried out, 0 out of 13 were left standing. This shows that none of the Memcached servers located in Greece responded to UDP. Turning off the UDP port for Memcached had been a widely deployed security measure following a sequence of devastating attacks exploiting this protocol [7].

Augumenting DNS dataset. To ensure our DNS dataset also has a representative sample of authoritative servers, we decided to retrieve top Greek domains from the list provided by TUM (see Section IV). We parsed enough of the list to obtain a good amount of domains. We obtained 1,023 Greek domains. We queried our local resolver for "NS" records on the Greek domains to find authoritative nameservers. We then performed a DNS query for the nameserver domains to resolve their IP addresses. As a final check, we made sure that the IPs were in Greece's IP range by using ipinfo [42], an insightful geolocation tool for IPs. Thus, we collected 926 authoritative nameservers and the domains each one was authoritative for.

Collect amplification factors. Secondly, we wish to rank the filtered hosts based on the BAF (see Equation (1)). For this purpose, we again hand-crafted special packets for each protocol. Having the attacker's reasoning in mind, we wanted to create the packets that would lead to the highest amplification factor. Appendix C shows other details (e.g. code) related to these packets.

DNS. An adversary aims to make the query size smaller, i.e. query a smaller domain, while getting substantial responses. This could happen if the attacker creates a short domain with a large "TXT" record, for instance. However, we assumed that an attacker would like to leave as few traces as possible and use existing domains. Since there are millions of domains, we restricted ourselves to rank only the top-level domains (the list retrieved from IANA mentioned in Section IV) according to their BAF. We used the well-established "ANY" query for the following measurements, as attackers typically used it in DNS-based amplification attacks too [31] since nameservers (if they support this query type) are expected to return all RRs of the domain in the query.

The TLDs are among the shortest domains one could use in a query. Surprisingly, ".sl" (the country-code top-level domain for Sierra Leone) returned the highest BAF of 177.83. This was an extreme value and is also a theoretical one as the response size had 5,513 bytes (larger than the maximum UDP size of 4,096) and was sent over TCP. This motivated us enough that the domain contained many records and was a worthy candidate for the query. Note that this measurement was done on all TLDs using the Google Public DNS recursive resolver (8.8.8.8) to ensure reproducibility [43]. We then proceeded to query the DNS hosts with the "ANY" query on the ".sl" domain, with an EDNS0 record setting the buffer size to 4,096 and the "DO" bit

set to 1 (to request a DNSKEY signed answer), following the methodology of [31], as we were interested in getting the maximum response. We got a BAF as high as 132.19, which is still very attractive for an attacker. This query was expected to yield small amplification factors for the authoritative nameservers, so we performed the same query on a different domain instead. We made one request per domain for which the nameservers were authoritative and kept the maximum BAF attained.

NTP. Alongside the notorious "monlist" (mon_getlist_1), we also sent three other Mode 7 (private) packets, namely "peer_list", "peer_list_sum" and "get_restricted", as they were also known to lead to good amplification factors [44]. The "peer_list" request will return all hosts with whom a server is peering, "peer_list_sum" returns the same information as the previous request but also includes some metadata about the peers. Lastly, a "get_restrict" response will contain hosts to whom firewall rules are applied (e.g., allowlisted IPs or blocklisted). If a server responds to the last query, this also leads to an information disclosure vulnerability attack.

Memcached. Following the 1.3Tbps DDoS Attack on GitHub [45], Akamai disclosed the type of queries attackers sent to public Memcached servers [29]. Even though none of our servers responded to UDP requests, we decided to still work on the methodology to send a packet that is expected to get a significant response from Memcached servers. We also validated the methods on an open Memcached server from one of our peers from France. It is important to note that Memcached operates in its essence in a very similar fashion to a key-value store. Users can query using a "get <key>" command to retrieve the value assigned to the key. Typically, the maximum size of a value is 1 megabyte (MB). An attacker can use this command on an existing key (that is short) associated with a considerable value. However, maximum effectiveness is achieved when the attacker sets the smallest key possible (with the size of 1 character) to the largest value possible (1 MB of data). After this, the "get" command can be sent alongside the key inserted previously. An attacker can even chain the keys, sending "get $< key > < key >$ ...", which would lead to a tremendous response. This type of packet was observed in [29]. The strategy will lead to an amplification factor in tens of thousands, which is unprecedented compared to the other protocols studied. We decided to work with whatever was stored on the server and compute the BAF for a "get" request with a single key, namely the one with the largest associated value (so without implanting a key and value pair). More details and our implementation of this approach are presented in Appendix C.

Gather factors. To investigate which servers achieve a large or small BAF adjacent to our measurements, we also collect metadata that will later be analysed in conjunction with the amplification factor. This has been done to paint a better picture of what leads to a higher amplification factor. The code that supports the collection of factors can be seen in Appendix D.

For DNS, we collected the software run by hosts by fingerprinting using "fpdns" [46]. From the initial Censys data, we collected various information about the AS, such as the AS description and the AS number, as well as the vendor (i.e. Microsoft) and the product (operating system, i.e. Linux). By using the "dig" command [47], a flexible DNS lookup utility tool, we also retrieved what EDNS0 buffer size a DNS server was advertising (if any). For NTP, we fetched the system variable (i.e. operating system) of each host by using the "ntpq" utility tool [48].

TABLE I: BAF per protocol. "All" represents the mean BAF across all open amplifiers, 50% and 10% only consider the most vulnerable 50% or 10% of all open amplifiers. In the third column, in the NTP row, the first number shows the number of servers that responded to a Mode 3 request, and the second the hosts that responded to "monlist".

Protocol	# of hosts	# of open hosts	BAF			Strategy
			All	50%	10%	
DNS	7532	2398	16.5	33.1	128.5	"ANY" on ".sl"
DNS_{auth}	926	926	11.8	18.4	26.0	Max. "ANY" on own domains
NTP	25962	25129 / 632	2.8	4.7	19.1	Max. of private queries
Memcached	13	0	0	0	0	"Get" request

Vulnerability to looping attacks. Besides the BAF measurement, we analyse the same protocol pairs for DNS and NTP, which are vulnerable to looping pairs. To do this, we followed the methodology and ran the same experiments as Pan et al. [16]. We forked the repository in order to run the code (after making some small changes for the scale of our experiment) on our datasets [49]. The procedure they set up is composed of 5 steps, which are detailed in Appendix G. Due to the difference in size of the datasets (we had around 100 times fewer hosts), we had to adapt the code and change some of the authors' decisions. After the second step, they find 90 clusters with at least $10,000$ responses, and they only sample packets from these clusters in step 3. In our experiment, we considered all the clusters with at least 1 response. Furthermore, in step 4, the authors only consider edges with at least 100 participants, whereas we kept all edges with at least 2 participants (not 1, to avoid self-loops). We have also skipped running the last stage, which would have involved setting up a proxy. If this had been done incorrectly, we would have risked starting an actual loop between two servers.

VI. Responsible Research

This section will present the considerations we made when conducting our experiments to follow an ethical research approach.

Ethical Considerations. When gathering initial information about hosts, we employed passive scanning to be as non-intrusive as possible. In contrast, if done without proper authorisation, active scanning could have been seen as malicious by the other parties. Furthermore, in the filtering and measurement stage of our methodology, described in Section V, we send packets to a host once, and we set a cool-down period of 100 milliseconds between requests to prevent flooding the target with requests, as well as not exhausting the bandwidth of any network links, respectively.

We will also make sure not to publish the IP addresses of hosts that are amplifiers to avoid the possibility of an attacker using this information and launching a DDoS attack. We wish to refrain from facilitating anyone with the resources to carry out such an attack.

Reproducibility. Due to the dynamic character of the world's networks, it is impossible to guarantee the complete reproducibility of our experiments. For instance, due to IP address churn, the hosts we gathered might have different IPs if collected at different times in the future [50]. Similarly, the vantage point from which the experiments are conducted might influence the results [51]. Nonetheless, we have created a Dockerfile that specifies everything one needs to run our experiments in the same setting (i.e. the same dependencies and versions). The scripts are also open-source and hosted on the author's GitHub [52].

VII. Results

Amplification attacks. The amplification factors we achieved using the three protocols are shown in Table I. Note that for

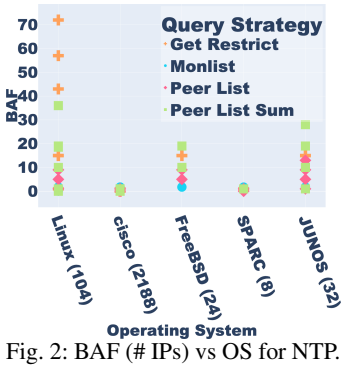

Fig. 2: BAF (# IPs) vs OS for NTP.

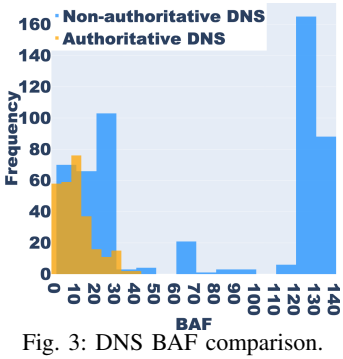

Fig. 3: DNS BAF comparison.

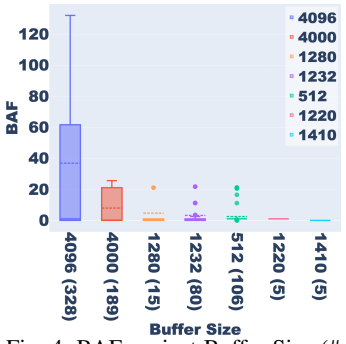

Fig. 4: BAF against Buffer Size (# IPs).

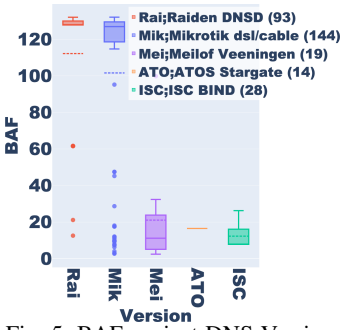

Fig. 5: BAF against DNS Version. The # of IPs per group is shown in the legend.

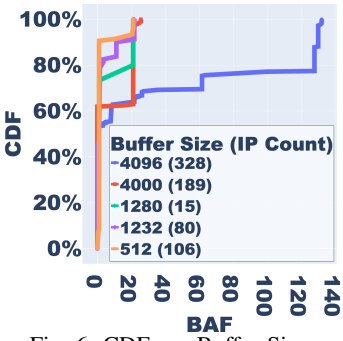

Fig. 6: CDF per Buffer Size.

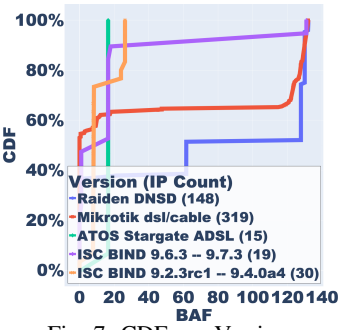

Fig. 7: CDF per Version.

Memcached, none of the servers were open to UDP requests. Similarly, we did not find significant amplification factors for NTP, as observed previously [7]. We attribute this to the effective campaign against private queries (specifically "monlist") done by [26]. Note that versions before 4.2.7p26 were vulnerable to traffic amplification via the "monlist" request [53]. Still, as Fig. 2 shows, some servers still respond to Mode 7 queries, which should not be the standard. Many hosts running Cisco OS have an amplification factor of 0, indicating they are well-configured. In contrast, Linux machines obtain the highest BAFs observed, likely due to misconfiguration. A detailed breakdown of the distribution of BAFs for NTP can be seen in Fig. 13 in Appendix E. We have no quantitative results for Memcached as none of the servers responded to UDP, but a thorough discussion about what BAF could theoretically be achieved is presented in Appendix C.

The story, however, is much different for DNS. This protocol can be weaponised to a large extent in the context of amplification attacks. The worst 250 DNS amplifiers located in Greece cumulatively achieve an amplification factor of $32,087$. A histogram of the amplification factor of DNS servers is presented in Fig. 3, which also shows that authoritative nameservers are more difficult to weaponise. Approximately 11% of all open DNS servers achieved a BAF greater than 100. This should prove that such effective amplifiers would be relatively easy for an attacker to find in the wild. We further examined the relationship between the amplification factor and potential causes. The distribution of amplification factors can be seen in Fig. 12 in Appendix E.

1. EDNS0 buffer size. The introduction of the EDNS0 extension made DNS responses larger than 512 bytes possible [54]. Clients can specify the maximum response size they can handle (rclass in the header), and similarly, hosts can optionally advertise the largest response size they accept [55]. After collecting the buffer size of our DNS hosts, we wanted to see if there was a positive correlation with the amplification factors, i.e., hosts that advertise a larger buffer size also attain a larger BAF.

We observed that the hosts (whose buffer size we knew) that achieved the most significant amplification factors published a buffer size of $4,096$ (Fig. 4), where the median is situated around the value of 40. The CDF in Fig. 6 also shows the noticeable amplification factors for the group that advertised a buffer size of $4,096$, $\approx 25\%$ of them achieve a BAF ≥ 100 (around 82 hosts). The difference to the other groups is apparent, as the other 4 groups get a BAF of 20 at most. Note that we plotted the top 5 groups according to the mean BAF and only kept groups with at least 10 members.

There have been attempts to discourage setting this considerable buffer size value since it can lead to IP Fragmentation [56]. It has been strongly advised to set it to a value such as $1,232$, which should be enough for all practical purposes, as suggested by DNS flag day 2020 [56], which was a collaborative initiative by DNS software and service providers to promote better DNS protocol compliance. Furthermore, a large buffer size significantly increases the amplification potential, making the problem two-fold. We have also observed cases where the buffer size is not correctly configured, i.e. a server advertises a value but does not truncate and switches to TCP when receiving a more extensive response. This happened, for instance, in the group that selected a buffer size of 512, then technically, the maximum attainable BAF should have been $\frac{512}{31} \approx 16.5$, but we can see in Fig. 4 that there is a small subset of hosts that achieved a BAF of ≈ 20.

We want to underline that $\approx 45\%$ of the hosts whose buffer size we know advertised a value of 4,096, and $\approx 70\%$ advertised a value $\geq 4,000$. These results are similar to the ones found by Van Rijswijk-Deij et al. [57], who found that the prevalent configuration (in 90% of the servers that adopted EDNS0) for the buffer size was a value at least as high as 4,000 bytes. We obtained a slightly smaller result, likely due to a number of hosts adopting the DNS flag day recommendation; still, the vast majority advertised a large value for the buffer size (i.e. $\geq 4,000$).

2. DNS implementation. We found two DNS versions (after running "fpdns") which achieved very high BAFs, namely Raiden DNSD and MikroTik. In Fig. 5, we can see the mass of the two implementations centred around the substantial value of 120. Note that we have ignored hosts with BAFs ≤ 1 in this figure, as they are not considered amplifiers. With a closer inspection of the distribution of all hosts running the respective DNS software, we observe from the cumulative distribution function in Fig. 7 that $\approx 50\%$ of all Raiden DNSD hosts have a BAF larger than 120. Similarly, the figure shows that $\approx 35\%$

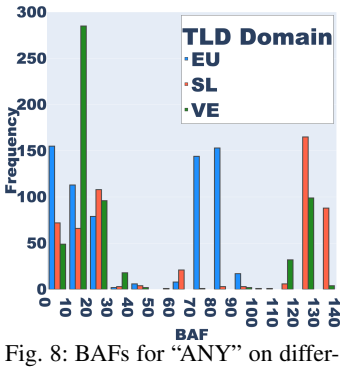

Fig. 8: BAFs for "ANY" on different domains.

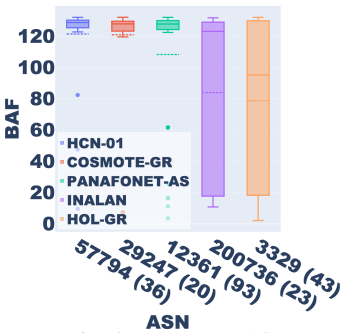

Fig. 9: BAFs per AS.

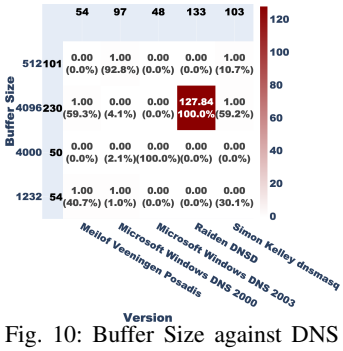

Fig. 10: Buffer Size against DNS software.

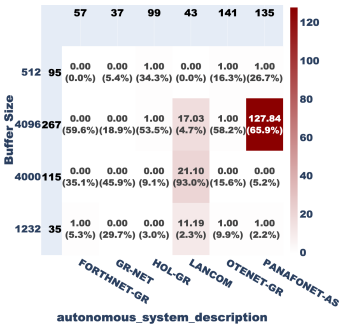

Fig. 11: Buffer Size against AS.

of MikroTik hosts have a BAF ≥ 120. This amounts to around 74 Raiden and 111 MikroTik hosts with a BAF greater than 120. Since Fig. 7 shows the top 5 groups according to the mean BAF (and whose group count is ≥ 10), we see that there is a significant difference compared to the other 3 versions that are part of the top 5, ATOS Stargate and BIND 9.2.3rc1 - 9.4.0a4 can get at most a BAF of ≈ 20. Only $\approx 10\%$ of hosts running BIND 9.6.3 - 9.7.3 obtain a BAF comparable to MikroTik and Raiden, i.e. ≥ 120. However, this is still a tiny amount of servers compared to the other two (around 9). Furthermore, looking at the top 250 hosts according to the BAF obtained (regardless of whether we know their DNS implementation or not), we find that 175 of them are running either MikroTik or Raiden DNSD, that is 70% of the top 250 hosts. The other 30% (i.e. 75) of hosts are made up of 73 whose software we could not find, and the remaining two are running two different versions of BIND [58].

We further investigate hosts that run one of these two DNS implementations to determine the root cause of their predisposition to high amplification. For Raiden, we have not been able to reach a meaningful conclusion using information from their website [59]; however, plotting a heatmap of the buffer size advertised versus the DNS implementation in Fig. 10, we notice that all of the Raiden servers that advertised a buffer size did so with a value of 4,096, which leads us to think that this is the preconfigured setting for the Raiden DNS implementation.

For MikroTik hosts, we could not retrieve the buffer size empirically; thus, it is not present in the same heatmap. However, MikroTik DNS documents that the buffer size it uses by default is $4,096$ [60]. This leads us to assume that this default value and is likely not overridden by administrators.

3. Domain. Unsurprisingly, the domain queried plays a crucial role in the size of DNS responses. This is an area that an attacker might specifically optimise, i.e. choosing the best domain given a specific nameserver. Three TLDs (.eu, .sl and .ve) and the BAFs achieved are compared in Fig. 8. The figure shows that "ANY" gives the highest amplification for many hosts for the ".sl" TLD, whereas .ve also obtains high BAFs but with a much smaller frequency; we can see a peak in frequency for a BAF between 20 and 30. The ".eu" TLD obtains smaller BAFs, around 70 to 80, for a decent number of servers. Overall, .sl reliably gets very high BAF values (i.e., for numerous subsets of hosts). Note that many other domains might get comparable BAFs, and an attacker might try them all for each DNS server and only use the one that rewards them with the largest amplification factor. A complete measurement study of domains and their relationship to response size was done by Van Der Toorn et al. [31].

TABLE II: DNS and NTP Loops.

Protocol	Cycle	# of Involved IPs	In-Min IP	Edge Amount	# of Pairs
DNS	[49, 49]	3	3	3	
DNS	[3, 3]	3	3	3	
NTP	[2, 2]	2	2	1	

Number of all affected IPs (DNS): 5
Number of all affected IPs (NTP): 2

4. AS. Looking at the networks where hosts are situated, we noticed that some ASs are more vulnerable than others, as depicted in Fig. 9. Note that this does not imply something inherently wrong with these networks, but rather that there are vulnerable hosts with likely misconfigured settings (such as the EDNS0 buffer size), see in Fig. 11. We note that the first 3 ASs have most of their mass concentrated around the value of 120. Additionally, out of the top 250 hosts, according to their BAF, 123 of them are in one of the three ASs. In Fig. 11, we see a heatmap of the buffer size against the AS description, and we can observe that the third AS from Fig. 9 achieves the highest median BAF of 127.84, and 65.9% of its hosts advertise a buffer size of 4,096. This heatmap should further prove that there is a clear tendency to obtain a higher BAF if a larger buffer size is advertised.

5. Product and Vendor. We have noticed that 99% of hosts running RouterOS (from the vendor MikroTik) use the MikroTik DNS software but achieve very small BAFs, as it can be seen in Fig. 16 in Appendix F, where the median BAF of this group is 0. Interestingly, two other groups run the same DNS software and tend to get a high median amplification factor, namely 121.94 and 120.92 for hosts running DSM (from the vendor Synology) [61] and servers running Linux, respectively. This ties in with the information that MikroTik DNS servers may advertise a buffer size of $4,096$ by default. Hosts that run RouterOS and MikroTik DNS are not affected by the large default buffer size, likely due to other cautionary measures that might have been implemented, such as answering with a HINFO RR to "ANY" queries, as has been advised by RFC8482 to provide minimally sized responses to "ANY" [32]. More details about the BAFs achieved by different vendors and products are presented in Figs. 14 and 15.

Looping Attacks. Our hosts are vulnerable to loopy attacks, as we found a few loops among the same protocol server pairs. A summary is shown in Table II. Note that we have not analysed cross-protocol vulnerabilities.

VIII. Discussion

We have noticed that "ANY" and the EDNS0 buffer size of 4,096 are the main culprits of DNS amplification attacks. We

have not found significant amplification potential for NTP and Memcached, likely due to effective patching campaigns held in the past [26]. This does not imply that these protocols are not dangerous around the world. Thus, we still recommend patching NTP to turn off private queries (which is the case for "ntpd" versions of 4.2.8 and above [62]) and for Memcached to disable the UDP port by default [63], by upgrading the vulnerable version to version "1.5.6" or above [64].

Peer outcomes. The influence of the EDNS0 buffer size on BAF in the context of DNS amplification spans other countries as well, namely Belgium (BE), Luxembourg (LU), France (FR), Sweden (SE) and the Netherlands (NL). However, the distribution of which values for the buffer size are advertised differs from country to country; for instance, hosts located in NL choose a size of 1,232 way more often than a value larger than 4k (almost $5\times$ more often), showing a better adoption of the measures suggested by [56]. In our case, a value larger than 4k was advertised more than $6\times$ compared to 1,232. We also see 1,232 being advertised in the majority of cases in FR. The BAFs we found for DNS are similar to the ones observed in BE, LU and SE. Moreover, recursive NS are more easily weaponisable in these countries than authoritative NS. When analysing NTP and Memcached amplification, FR and NL are more vulnerable than any other country from our peers, with hosts that run these protocols and lead to high BAFs. For FR, we conclude that this might be the case due to the sheer number of hosts, the likelihood of finding exploitable hosts being thus larger. Lastly, BE, FR, and SE also showcase the Linux NTP servers, which leads to large BAFs via Mode 7 queries.

We found similar results to what others have discovered in previous work [57]; they observed that the majority of hosts that adopted EDNS0 advertise a buffer size of at least 4,000 bytes, namely in 70% of our hosts. We noted the tendency for the BAF to increase as the buffer size becomes larger. Another interesting finding was that we even received larger responses than the advertised buffer size, which should not be the norm. As RFC9210 [65] describes, when the data does not fit within the buffer size, the responding server should set the truncated flag to true in its response, and the requesting server should retry over TCP. Thus, the size of the answer should not exceed the buffer size. We concluded that two DNS implementations, MikroTik DNS and Raiden DNSD, set a default buffer size value of 4,096. Hosts running these two also got the highest BAFs out of all our DNS hosts, making up 70% out of the top 250 of our hosts. Reflecting on our DNS results, we want to highlight that BAFs between 120 and 130 are readily achievable by attackers since relevant sources cite smaller values for DNS amplification. For instance, RFC5358 [37] states that values of 80 can be achieved, and CISA (Cybersecurity and Infrastructure Agency) mentions DNS can reach values between 28 and 54. The traffic that the top DNS hosts can generate can be destructive; an attacker capable of generating 100 megabits/second in queries can generate traffic as high as 3.89 terabits/second at the victim, as showcased by Table III. This puts the DNS amplification provided by the top hosts located in Greece in a class of its own compared to the impactful Memcached attack from 2018, which reached a peak traffic of 1.3 Tbps [29].

Mitigations. As effective campaigns performed in the past managed to mitigate the dangers of NTP [26] and Memcached [29], we hope that the cybersecurity community will continue to hold similar initiatives for DNS. We propose restricting the use of "ANY" queries as they are bound to lead to high amplification, which has also been suggested by an RFC [32]. Even if extensive records need to be sent, we advise following the DNS flag day 2020 recommendation of setting the EDNS0 buffer size to 1,232 and adequately configuring it to send larger responses over TCP. Finally, we want to encourage every ISP to implement BCP38 [8] to fight IP spoofing, which would lead to the extinction of spoof-based attacks.

TABLE III: Theoretical amplification traffic when the attacker has an uplink of 100 Mbps.

# of top hosts	Cumulative BAF	Victim (bits/s)
100	13,073.35	1.307 T
250	32,087.23	3.208 T
500	38,964.94	3.8964 T

Limitations. Our work has only explored three protocols, but many other protocols that could be exploited exist (such as CharGen or SNMP [7]). Furthermore, our results should serve as a lower bound to what an attacker could achieve in reality, as an attacker might further optimise and achieve a higher BAF for hosts than we achieved a small one. We have also only analysed the IPv4 range, as there were no IPv6 hosts on Censys running one of the studied protocols. Still, we want to highlight the need for further studies for IPv6 hosts. Lastly, a worldwide study would have to be conducted to confirm or contradict patterns found in our work, such as the influence of the EDNS0 buffer size.

IX. Conclusion

Summary. We have analysed the amplification potential of three protocols from hosts located in Greece. We have shown that DNS remains a very effective amplification vector; the cumulative BAF provided by the worst 250 DNS amplifiers was $\approx 32,000$. We have determined that "ANY" queries and a large EDNS0 buffer size lead to high DNS amplification. Moreover, two popular DNS implementations (Raiden DNSD and MikroTik) set a default value of 4,096 for the buffer size. In contrast, we have not found significant amplification potential in NTP and Memcached hosts (located in Greece). Lastly, we discovered that looping attacks are also possible among the hosts considered. We found some vulnerable same protocol server pairs, leading to 7 potential loops.

Future Work: We would like to raise awareness of the dangers of DRDoS attacks and the need for further research. While our measurement study focused on three protocols, numerous other protocols (e.g. CharGen, SNMP) could be exploited. Future research should expand to include these and potentially other protocols to provide a more comprehensive understanding of amplification threats. This should be done in the context of an extensive and diverse dataset with better geographic coverage to ensure that the findings are globally representative and account for regional differences in network configurations and security practices. Since our results are a lower bound of what attackers can accomplish in practice, we believe studying amplifiers and real DRDoS attacks could give invaluable insights into how protocols are exploited (i.e. how an attacker optimises to reach the highest BAF). Last but not least, our analysis was limited to the IPv4 range. Further research should focus on identifying and analysing IPv6 hosts running protocols that can lead to amplification to understand this risk in the IPv6 space. This is crucial as the adoption of IPv6 continues to grow.

References

[1] "What is a denial-of-service (DoS) attack?" Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/glossary/denial-of-service/.

[2] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004. [Online]. Available: https://dl.acm.org/doi/10.1145/997150.997156

[3] "10 most common types of cyber attacks." CrowdStrike, https://www.crowdstrike.com/cybersecurity-101/cyberattacks/most-common-types-of-cyberattacks/.

[4] "What is a DDoS attack?" Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack/.

[5] "What is a Botnet?" CrowdStrike, https://www.crowdstrike.com/cybersecurity-101/botnets/.

[6] "What Is The CIA Triad? — Confidentiality, Integrity And Availability," SentinelOne, https://www.sentinelone.com/cybersecurity-101/cia-triad/.

[7] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *Proceedings 2014 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2014. [Online]. Available: https://www.ndss-symposium.org/ndss2014/programme/amplification-hell-revisiting-network-protocols-ddos-abuse/

[8] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." RFC Editor, RFC 2827, 2000. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2827.html

[9] "Spoofer Project." CAIDA, https://www.caida.org/projects/spoofer/.

[10] "NTP amplification DDoS attack." Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/ntp-amplification-ddos-attack/.

[11] "The first DDoS attack was 20 years ago. this is what we've learned since." https://www.technologyreview.com/2019/04/18/103186/the-first-ddos-attack-was-20-years-ago-this-is-what-weve-learned-since/.

[12] "Famous DDoS attacks — the largest DDoS attacks of all time." Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/famous-ddos-attacks/.

[13] "DDoS threat report for 2024 Q1." Cloudflare, https://blog.cloudflare.com/ddos-threat-report-for-2024-q1/.

[14] "DDoS for hire." Imperva, https://www.imperva.com/learn/ddos/booters-stressers-ddosers/.

[15] "What is an autonomous system?" Cloudflare, https://www.cloudflare.com/en-gb/learning/network-layer/what-is-an-autonomous-system/.

[16] Y. Pan, A. Ascheman, and C. Rossow, "Loopy Hell(ow): Infinite Traffic Loops at the Application Layer," Apr. 2024. [Online]. Available: https://publications.cispa.de/articles/conference_contribution/Loopy_Hell_ow_In_nite_Traffic_Loops_at_the_Application_Layer/25470952

[17] "What is a DNS Amplification DDoS Attack?" Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/dns-amplification-ddos-attack/.

[18] "What is an NTP Amplification DDoS Attack?" Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/ntp-amplification-ddos-attack/.

[19] "What is a Memcached DDoS Attack?" Cloudflare, https://www.cloudflare.com/en-gb/learning/ddos/memcached-ddos-attack/.

[20] "SNMP Reflection Attacks." Imperva, https://www.imperva.com/learn/ddos/snmp-reflection/.

[21] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[22] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[23] "What is DNS?" Cloudflare, https://www.cloudflare.com/en-gb/learning/dns/what-is-dns/.

[24] "The evolving DNS threat landscape." Cloudflare, https://www.cloudflare.com/en-gb/the-net/dns-landscape/.

[25] D. Mills, "Network Time Protocol (NTP)," RFC Editor, RFC 958, 1985. [Online]. Available: https://www.rfc-editor.org/rfc/rfc958

[26] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 111–125. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kuhrer

[27] "Memcached Repository." GitHub, https://github.com/memcached/memcached.

[28] "Memcached website." Memcached, https://www.memcached.org/.

[29] "State of the Internet / Security: Summer 2018 Attack Spotlight." Akamai Technologies, https://www.akamai.com/site/en/documents/state-of-the-internet/soti-summer-2018-attack-spotlight.pdf.

[30] H. Griffioen, K. Oosthoek, P. Van Der Knaap, and C. Doerr, "Scan, Test, Execute: Adversarial Tactics in Amplification DDoS Attacks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 940–954. [Online]. Available: https://dl.acm.org/doi/10.1145/3460120.3484747

[31] O. Van Der Toorn, J. Krupp, M. Jonker, R. Van Rijswijk-Deij, C. Rossow, and A. Sperotto, "ANYway: Measuring the Amplification DDoS Potential of Domains," in *2021 17th International Conference on Network and Service Management (CNSM)*. Izmir, Turkey: IEEE, Oct. 2021, pp. 500–508. [Online]. Available: https://ieeexplore.ieee.org/document/9615596/

[32] J. Abley, O. Gudmundsson, M. Majkowski, and E. Hunt, "Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY." RFC Editor, RFC 8482, 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8482

[33] "Top-level domains." IANA, https://data.iana.org/TLD/tlds-alpha-by-domain.txt.

[34] NTP Pool Project, https://www.ntppool.org/en/.

[35] "Top domains." TUM, https://toplists.net.in.tum.de/archive.

[36] "Recursive vs Authoritative DNS — What's the difference?" NSLookup.io, https://www.nslookup.io/learning/recursive-vs-authoritative-dns/.

[37] J. Damas and F. Neves, "Preventing use of recursive nameservers in reflector attacks," RFC Editor, RFC 5358, 2008. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5358

[38] "SNTP overview." TimeTools, https://timetoolsltd.com/ntp/sntp-overview/.

[39] Wireshark, https://www.wireshark.org/.

[40] R. Rohith, M. Moharir, and G. Shobha, "SCAPY - a powerful interactive packet manipulation program," in *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*. IEEE, 2018.

[41] M. Perham, "Slabs, Pages, Chunks and Memcached." https://www.mikeperham.com/2009/06/22/slabs-pages-chunks-and-memcached/.

[42] IP Info, https://ipinfo.io/.

[43] "Public DNS." Google, https://developers.google.com/speed/public-dns/.

[44] J. Hart, "R7-2014-12: More Amplification Vulnerabilities in NTP Allow Even More DRDoS Attacks." Rapid7, https://www.rapid7.com/blog/post/2014/08/25/r7-2014-12-more-amplification-vulnerabilities-in-ntp-allow-even-more-drdos-attacks/.

[45] "GitHub Survived the Biggest DDoS Attack Ever Recorded." Wired, https://www.wired.com/story/github-ddos-memcached/.

[46] "DNS server fingeprinting tool ." GitHub, https://github.com/kirei/fpdns.

[47] "dig - Linux man page." die.net, https://linux.die.net/man/1/dig.

[48] "NTP Debugging Techniques." NTPsec, https://docs.ntpsec.org/latest/debug.html.

[49] "Cispa Loopy repository." GitHub, https://github.com/cispa/loop-DoS/tree/main.

[50] H. Griffioen and C. Doerr, "Quantifying autonomous system IP churn using attack traffic of botnets," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*. Virtual Event Ireland: ACM, Aug. 2020, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/3407023.3407051

[51] D. Wagner, D. Kopp, M. Wichtlhuber, C. Dietzel, O. Hohlfeld, G. Smaragdakis, and A. Feldmann, "United We Stand: Collaborative Detection and Mitigation of Amplification DDoS Attacks at Scale," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 970–987. [Online]. Available: https://dl.acm.org/doi/10.1145/3460120.3485385

[52] R. Toader, GitHub, https://github.com/RaresToader/cse3000.

[53] "CVE-2013-5211 Detail." NIST, https://nvd.nist.gov/vuln/detail/CVE-2013-5211.

[54] J. Damas, M. Graff, and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))." RFC Editor, RFC 6891, 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6891

[55] "What's EDNS All About (And Why Should I Care)?" The DNS Institute, https://dnsinstitute.com/documentation/dnssec-guide/ch03s05.html.

[56] "DNS flag day 2020." DNS Flag Day, https://www.dnsflagday.net/2020/.

[57] 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*. Vancouver BC Canada: ACM, 2014, pp. 449–460. [Online]. Available: https://dl.acm.org/doi/10.1145/2663716.2663731

[58] "BIND 9." Internet Systems Consortium,https://www.isc.org/bind/.

[59] "Raiden DNSD," Raiden, http://www.raidendnsd.com/.

[60] "Router OS DNS documentation." MikroTik, https://help.mikrotik.com/docs/display/ROS/DNS.

[61] "DiskStation Manager 7.0.1." Synology, https://www.synology.com/en-us/DSM70.

[62] "4.2.8-series Changelog." NTP Project,https://www.ntp.org/support/securitynotice/4_2_8-series-changelog/.

[63] "How To Secure Memcached by Reducing Exposure." DigitalOcean,https://www.digitalocean.com/community/tutorials/how-to-secure-memcached-by-reducing-exposure.

[64] "Release Notes 1.5.6." Memcached Repository, https://github.com/memcached/memcached/wiki/ReleaseNotes156.

[65] J. Kristoff and D. Wessels, "DNS Transport over TCP - Operational Requirements." RFC Editor, RFC 9210, 2022. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9210

Appendix A
Censys queries

The queries used to retrieve hosts from Censys are shown below. One command was used for each protocol. Note that the number of pages used was large enough to fetch all hosts from Censys. Alternatively, one could set the "pages" flag to -1 to retrieve all the pages.

- DNS:

```
censys search 'services.service_name: DNS and services.port:53 and location.country: "Greece"' --per-page 100 --pages 77 > dns.json
```

Listing 1: DNS query.

- NTP:

```
censys search 'services.service_name: NTP and services.port:123 and location.country: "Greece"' --per-page 100 --pages 267 > ntp.json
```

Listing 2: NTP query.

- Memcached:

```
censys search 'services.service_name: MEMCACHED and services.port:11211 and location.country: "Greece"' --per-page 100 --pages 1 > memcached.json
```

Listing 3: Memcached query.

Appendix B
Filter queries

After gathering our hosts, we send simple queries to each one to see whether or not they are open to the public in our so-called "filtering" stage. Each query strategy (i.e. hand-crafted packet) is shown below. For DNS, we send a simple "A" query to resolve the IP address of google.com, as demonstrated in Listing 4. When filtering NTP hosts, we first send a normal Mode 3 (client) request that can be seen in Listing 5 and then send a "monlist" packet to the servers that responded to the previous query, as shown in Listing 6. We send a "stats slabs" request for Memcached hosts to filter them, as shown in Listing 7.

- DNS:

```
pkt = IP(dst=ip) / UDP(dport=53, sport=RandShort()) / DNS(rd=1, qd=DNSQR(qname="google.com"))
```

Listing 4: DNS "A" request.

- NTP:

```
pkt = IP(dst=ip) / UDP(dport=123, sport=RandShort()) / NTP(version=4, mode=3)
```

Listing 5: NTP request.

```
data = "\x17\x00\x03\x2a" + "\x00" * 4
# 0x17 for private mode, 0x00 for response,
     0x03 for version, 0x2a for monlist
pkt = (IP(dst=ip) / UDP(dport=123, sport=
    RandShort()) / Raw(load=data))
```

Listing 6: NTP "monlist" query.

- Memcached:

```
pkt = IP(dst=ip) / UDP(sport=RandShort(),
    dport=11211) / Raw(load="\x00\x01\x00\
    x00\x00\x01\x00\x00stats slabs\r\n")
```

Listing 7: Memcached "stats slabs" request.

Appendix C
Measurement queries

To measure the BAF for the protocols, we have hand-crafted different packets that will lead to high amplification. In the case of DNS, we send an "ANY" request to the ".sl" TLD, presented in Listing 8. Note from the code that the request can easily be changed to be a "DNSKEY" or "RRSIG" request by changing the "query_type" accordingly. We have mostly tried "ANY" as it would lead to the highest amplification factor if the server responded to it. For NTP, we send a "monlist" request, which was presented in Listing 6, along with some other private Mode queries [44], that are shown in Listing 9. For Memcached, we need to send two initial queries to find information about the keys and values present in the server. Our first query is a "stats items" query (Listing 10). This command retrieves statistical information about the items stored in the cache, providing details about each slab class (a logical grouping of cache entries of similar size). We parse the output from this command to retrieve all slab IDs. We then send, for each slab ID, a "stats cachedump" request (Listing 11). The 0 in the request indicates no limit, i.e. the server should answer with all items. This query retrieves information about items stored in a specific slab class. Each entry (i.e. item) contains the key, the size of the associated value and a timestamp (when the item was last accessed or possibly when it was set, depending on the version and configuration). This is the information we were looking for as we look to find the key and value that would lead to the highest BAF for the request we are considering (i.e. "get $< key >$"). The code for this request is shown in Listing 12.

We also wanted to analyse the maximum theoretical BAF for Memcached for the "get $< key >< key > ... < key >$" request. We created a function for the theoretical BAF, named b, shown in Equation (2), where k is the size of the key, v is the size of the associated value, and t is the number of times the key appears in the "get" request. As t tends to infinity, the BAF becomes equal to $\frac{v}{k+1}$, as shown in Equation 3. In the ideal case, the key has the smallest size possible, one character (thus one byte), and the value the largest size possible, 1 MB (i.e. 10^6 bytes). This means that the largest theoretical BAF for this request will tend to a value of $\frac{10^6}{1+1} = \frac{10^6}{2} = 500k$. This should show that the size of the key and the value associated with it directly impact the BAF potential of Memcached.

$$b(k, v, t) = \frac{v \cdot t}{13 + t \cdot (k + 1)} \tag{2}$$

$$\lim_{t \to \infty} b(k, v, t) = \lim_{t \to \infty} \frac{v \cdot t}{13 + t \cdot (k + 1)} = \frac{v}{k + 1} \tag{3}$$

- DNS:

```
dns = DNS(ad=1)
dns.qd = DNSQR(qname=domain, qtype=
    query_type, qclass="IN")  # query
    section
dns.ar = DNSRROPT(rclass=4096, z=1)  #
    additional records
request = IP(dst=ip) / UDP(dport=53, sport=
    RandShort()) / dns
```

Listing 8: DNS "ANY" request on ".sl".

- NTP:

```
packet = IP(dst=target_ip) / UDP(dport=123,
     sport=RandShort()) / NTPPrivate(
        response=0,  # i.e. this is a
    request
        more=0,  # not expecting multiple
    packets
        version=2,  # version number 2/3
        implementation=3,  # assuming XNTPD
        auth=0,  # no authentication
        mode=7,  # 6 => mode 6 (control) or
     mode 7 (private)
        seq=0,  # sequence number
        request_code=command_code, # 0 for
    PEER_LIST, 1 for PEER_LIST_SUM, 16 for
    GET_RESTRICT
        # mode 6 => 12 for CTL_OP_REQ_NONCE
     and 31 for UNSETTRAP
        err=0,
        nb_items=0,  # number of data items
        mbz=0,  # must be zero
        data_item_size=0,  # size of each
    data item
    )
```

Listing 9: NTP private Mode requests.

- Memcached:

```
pkt = IP(dst=ip) / UDP(sport=RandShort(),
    dport=11211) /
Raw(load="\x00\x01\x00\x00\x00\x01\x00\
    x00stats items\r\n")
```

Listing 10: Memcached "stats items" request.

```
pkt = IP(dst=ip) / UDP(sport=RandShort(),
    dport=11211) /
Raw(load=f"\x00\x01\x00\x00\x00\x01\x00\
    x00stats cachedump {slab_id} 0\r\n")
```

Listing 11: Memcached "stats cachedump" request.

```
pkt = IP(dst=ip) / UDP(sport=RandShort(),
    dport=11211) /
Raw(load=f"\x00\x01\x00\x00\x00\x01\x00\
    x00get key\r\n")
```

Listing 12: Memcached "get $< key >$" request.

Appendix D
Collect metadata

To grasp the connection between certain factors and the BAF achieved by hosts, we had to augment our existing knowledge with some metadata. We had different procedures, the code of

which is shown below. For DNS, we retrieved the product (e.g. RedHat) and vendor information (e.g. Enterprise Linux) from the JSON file already provided by Censys. To classify an NS as recursive or not, we used the code snippet in Listing 13, where we essentially try to resolve the IP of "google.com" by using a specific nameserver recursively by setting the recursive flag. Note that we also set a timeout of one second and do not consider a server to be recursive in case of an error or a timeout. To obtain authoritative name servers, we used the flipped approach. We don't use any nameserver, but for each Greek domain we gathered, we looked up its authoritative nameserver via the "dig" query shown in Listing 14. As a result, we get the authoritative NS domains, which we then map to IP addresses with a simple "dig" query, see Listing 15. Finally, we only keep the authoritative NS IP if it is located in Greece, which we check using the geolocation tool "ipinfo" [42]; the code for this is shown in Listing 16. This outputs a JSON with various information, and we use the "country" attribute, which we match with "GR".

We have obtained the DNS software run by hosts by using the "fpdns" command line utility tool [46]. The code is presented in Listing 17. We set a timeout of 20 seconds per each request. Note that "fpdns" is not a well-maintained tool and likely has trouble distinguishing between newer versions of DNS software. We count this as a limitation of our analysis.

For NTP, we have only gathered the system information (i.e. the operating system run by a host). We used the "ntpq" CLI utility tool, as described in [48]. The code for this is shown in Listing 18. We parse the output and retrieve the value for the "system" attribute.

```
dig @<ip> google.com +rec +time=1
```

Listing 13: Resolve the IP of google.com recursively

```
dig <domain> NS +short
```

Listing 14: Finds the authoritative nameservers of the placeholder domain.

```
dig +short <nameserver_domain>
```

Listing 15: Resolve the IP address of the authoritative nameserver domain.

```
curl http://ipinfo.io/<ip_address>/json
```

Listing 16: Find geolocation information about placeholder IP.

```
fpdns -t 20 <ip_address>
```

Listing 17: Find DNS software information about the placeholder host.

```
ntpq -c rv <ip_address>
```

Listing 18: Find operating system ran by NTP host.

Appendix E
Piecharts and Boxplots

The distribution of DNS hosts according to their BAF is shown in Fig. 12. Most hosts are closed (68.2%); however, quite a large amount of DNS hosts (265) achieve a BAF $\geq$ 80. A similar visualisation for NTP is shown in Fig. 13. Most NTP servers (94.4%) did not respond to the "monlist" private

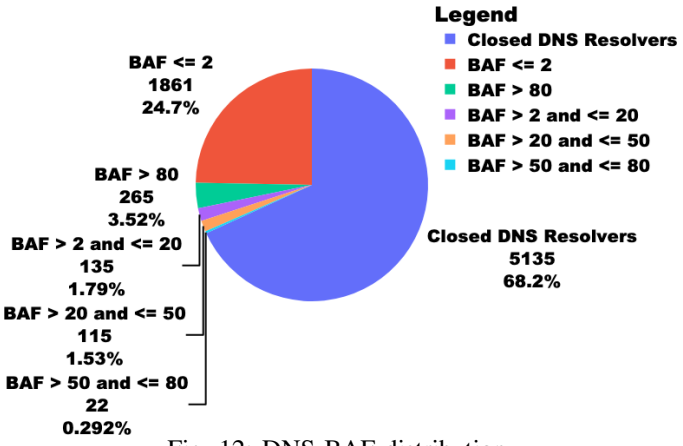

Fig. 12: DNS BAF distribution.

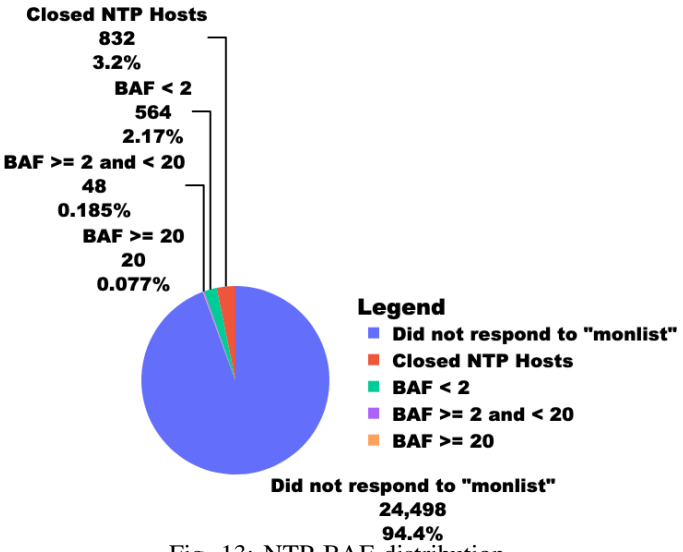

Fig. 13: NTP BAF distribution.

Mode query for NTP. A few servers are closed (3.2%), which was expected to ensure accurate and synchronised timekeeping across different devices and networks. Also, a tiny number (20) achieved a BAF $\geq$ 20.

The BAF distribution is depicted in a boxplot from the DNS hosts, whose vendor information we know, in Fig. 14. Red Hat hosts are secure, almost all achieving a BAF of 0. In contrast, MikroTik servers reach the largest BAFs and some Microsoft and Hikvision outlier servers. Furthermore, the product (operating system) boxplot shown in Fig. 15 also shows that servers running RouterOS (from MikroTik) and Linux achieve the highest BAF. Unlike this, Enterprise Linux hosts (from Red Hat) are secure, achieving very small BAFs. Note that there is a high correlation between Fig. 14 and Fig. 15 in the sense that, for instance, all the RedHat hosts are running Enterprise Linux or all Synology hosts are running DSM [61].

Appendix F
Heatmaps

The heatmaps from above compare pairs of factors, and each cell contains the median BAF attained per that respective group. The percentage of the feature on the x-axis is also displayed in parentheses. The total number of servers from certain features is displayed on the left and top borders for the y-axis and x-axis features. Fig. 16 shows that, interestingly, servers that get

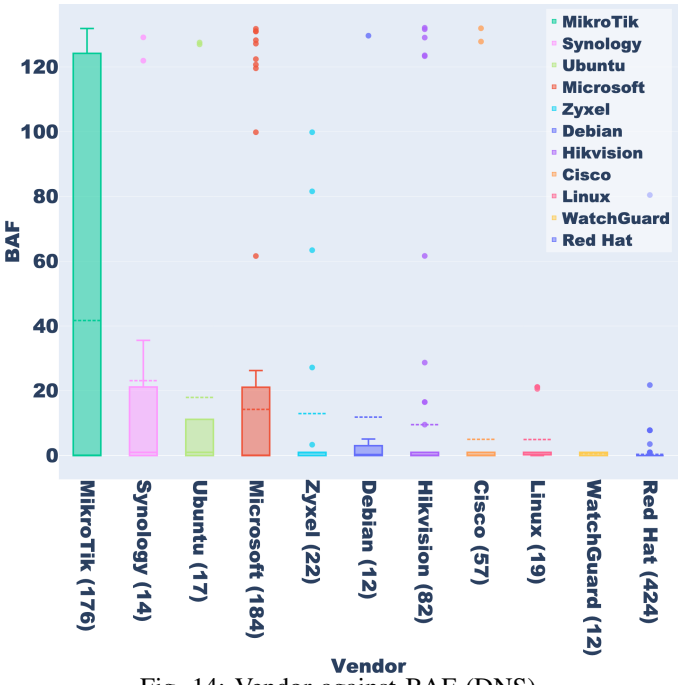

Fig. 14: Vendor against BAF (DNS).

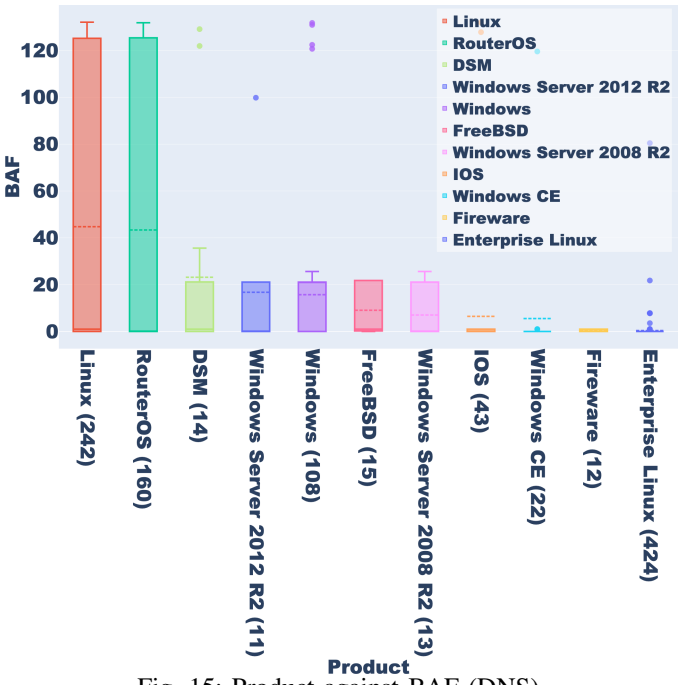

Fig. 15: Product against BAF (DNS).

a high BAF that use the MikroTik implementation are not the ones that run Router OS as their operating system. The ones that obtain a high BAF are servers that run DSM (DiskStation Manager - from Synology) and Linux. Similarly, the groups that get the highest BAF in Fig. 17 are the ones that advertise a Buffer Size of 4,096.

Appendix G
Loopy attacks

Below is a brief overview of the methodology created by [16] to find server pairs that are vulnerable to looping attacks.

Step 1. In the first stage discovery packets are sent to all

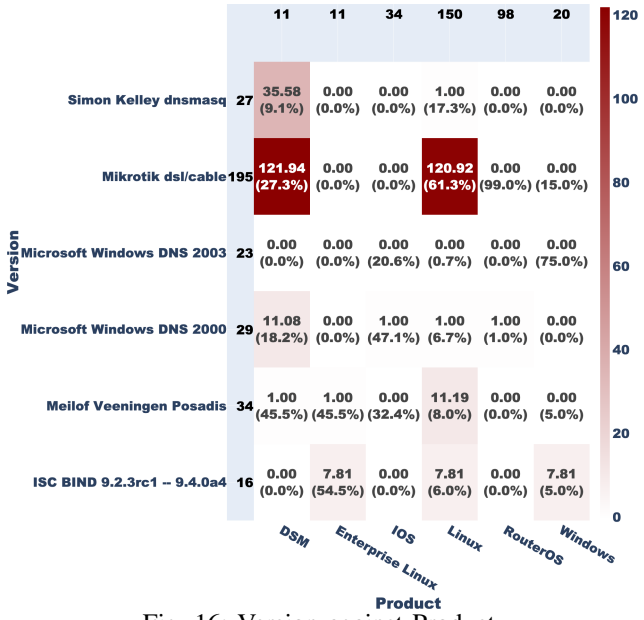

Fig. 16: Version against Product.

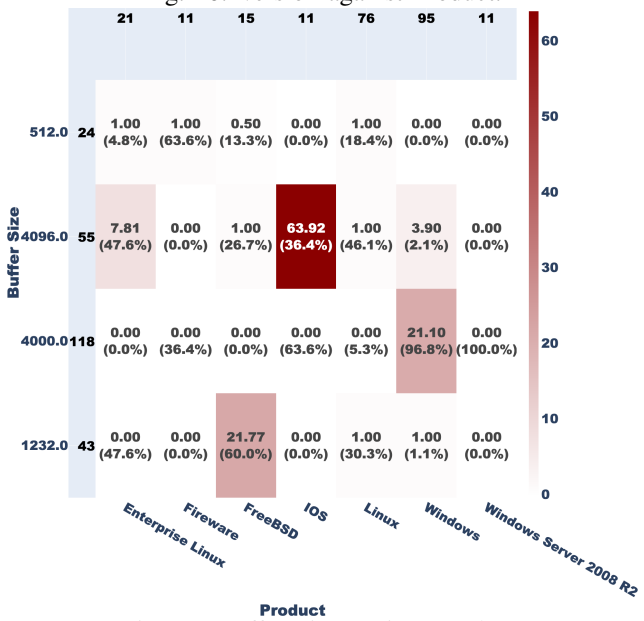

Fig. 17: Buffer Size against Product.

the servers go gather a complete set of responses. These could serve as the entry point to a traffic loop.

Step 2. Since there could be many distinct responses (depending on the number of hosts), the second stage clusters the responses based on their semantics. Insignificant syntactic differences, such as the transaction ID (TXID), are ignored since they cannot influence a DNS response. In the end, several clusters will hold semantically equivalent responses.

Step 3. With a similar reasoning as the second stage, in the third stage, random responses are sampled from each of the clusters. These will be the set of probes that will be sent again to each server. Sending all the initial responses to all the servers would have been both infeasible, due to the large number of initial responses, but also ineffective, since several packets from the same cluster are expected to have the same behaviour, and in turn, should themselves lead to the same response. The responses after this third stage also end up

being clustered in the same way as the second stage.

Step 4. With the information gathered in the third step, a loop graph can be formed. In this graph edges represent servers that reply to an input from one cluster with a response from the same or another cluster. Following this, a DFS is ran to find cycles with a length of at most 4.

Step 5. In the last stage, the loops that were identified previously are formally verified with a proxy. The proxy sits in between the two servers that are being tested and it forwards messages between the two. Once the proxy has forwarded enough messages, the two servers are validated as a loop pair. As this step required setting up a proxy, we have skipped it.