Amplification Hell: Revisiting Network Protocols for DDoS Abuse

Christian Rossow
VU University Amsterdam, The Netherlands
Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany
{firstname.lastname}@rub.de

***Abstract*—In distributed reflective denial-of-service (DRDoS) attacks, adversaries send requests to public servers (e.g., open recursive DNS resolvers) and spoof the IP address of a victim. These servers, in turn, flood the victim with valid responses and – unknowingly – exhaust its bandwidth. Recently, attackers launched DRDoS attacks with hundreds of Gb/s bandwidth of this kind. While the attack technique is well-known for a few protocols such as DNS, it is unclear if further protocols are vulnerable to similar or worse attacks.**

In this paper, we revisit popular UDP-based protocols of network services, online games, P2P filesharing networks and P2P botnets to assess their security against DRDoS abuse. We find that 14 protocols are susceptible to bandwidth amplification and multiply the traffic up to a factor 4670. In the worst case, attackers thus need only 0.02% of the bandwidth that they want their victim(s) to receive, enabling far more dangerous attacks than what is known today. Worse, we identify millions of public hosts that can be abused as amplifiers.

We then analyze more than 130 real-world DRDoS attacks. For this, we announce bait services to monitor their abuse and analyze darknet as well as network traffic from large ISPs. We use traffic analysis to detect both, victims and amplifiers, showing that attackers already started to abuse vulnerable protocols other than DNS. Lastly, we evaluate countermeasures against DRDoS attacks, such as preventing spoofing or hardening protocols and service configurations. We shows that carefully-crafted DRDoS attacks may evade poorly-designed rate limiting solutions. In addition, we show that some attacks evade packet-based filtering techniques, such as port-, content- or length-based filters.

I. Introduction

In a denial-of-service (DoS) attack, an attacker with financial, political or purely destructive motivation disrupts a service of a victim by adding an excessively high load to the victim's service(s). There are several forms of DoS attacks [30, 18], most of which are well-documented and used by attackers to disturb services for years. For example, attackers target victims with distributed DoS (DDoS) attacks and instruct infected PCs that are part of a malicious botnets [4]. Similarly, attackers can exhaust application-layer resources, such as the maximum number of database sessions of a web application [30].

NDSS '14, 23-26 February 2014, San Diego, CA, USA
 ISBN 1-891562-35-5
http://dx.doi.org/10.14722/ndss.2014.23233

In distributed reflective denial-of-service (DRDoS) attacks, an adversary aims to exhaust the victim's bandwidth. He abuses the fact that public servers of UDP-based network protocols respond to requests without further validating the identity (i.e., the IP address) of the sender. DRDoS offers many desired attack features for an adversary: (i) He disguises his identity, as victims receive backscatter traffic from *amplifiers*, i.e., systems that can be abused to send traffic to the victim on the attacker's behalf; (ii) The simultaneous abuse of multiple amplifiers permits a highly-distributed DoS attack to be conducted from a single Internet uplink. (iii) The traffic reflected to the victim is significantly larger in bandwidth than the traffic an attacker has to send to the amplifiers.

Adversaries abused the high potential of massive DRDoS attacks recently. In May 2012, attackers targeted a real-time financial exchange platform with a 167 Gb/s DRDoS attack [15]. In March 2013, attackers launched 300 Gb/s DRDoS attack against Spamhaus.org [16]. In August 2013, presumably politically-motivated attackers brought down GreenNet, an ISP hosting human rights groups, with a 100 Gb/s DRDoS attack [36]. In these known examples, attackers abused open DNS resolvers to amplify their attack traffic. The attacker(s) issued specially-crafted `ANY` requests to thousands of open resolvers and specified the victim's IP address as packet source. In turn, after successful name resolution, the resolvers sent several-kilobyte-large responses to the victim, exceeding its bandwidth capacity. With these attacks, DNS has been practically proven to be vulnerable to DRDoS abuse. One of the reasons for this vulnerability are the recent deployment of EDNS0 and DNSSEC, which significantly increases the DNS response sizes [5]. As a consequence, developers and administrators increasingly harden DNS servers against abuse, e.g., by closing millions of open resolvers and limiting the request rate per client [38]. However, DNS is not the only widely-deployed service, and little is known about angles for amplification of other popular network protocols.

In this paper, we revisit UDP-based network protocols and evaluate if they are vulnerable to amplification attacks. We identify and describe amplification vectors in 14 protocols of various services, including network services (such as NTP, SNMP, SSDP and NetBios), legacy services (CharGen and QOTD), peer-to-peer (P2P) filesharing networks (BitTorrent and Kad), game servers (Quake 3 and Steam) and P2P-based botnets. Many of these protocols have been designed decades ago way before DRDoS attacks emerged. As we show, attackers can abuse these protocols to multiply their attack bandwidth by factors from 3.8 (BitTorrent, NetBios)

up to 4670 (NTP). In combination with source IP address spoofing, this adds severe loads to designated DRDoS victims. For each protocol, we explore the set of potential amplifiers, for example, by scanning the Internet for open services, by crawling P2P networks, or by requesting game server lists from master servers. We show that millions of potential amplifiers are available for six vulnerable protocols – ideal for attackers that aim to distribute attacks over many traffic sources.

We then use a three-fold approach to understand if attackers abuse these vulnerable protocols. (i) We deploy bait services for these protocols and monitor how they are abused by attackers. (ii) We analyze if attackers scan for potential amplifiers by looking at the scanning noise of two darknets. (iii) We propose a light-weight method to detect DRDoS attacks via traffic analysis and deploy it at a large ISP. Our evaluation of more than 130 real-world DRDoS attacks shows that DNS is still the most popular DRDoS protocol, but we also already witnessed attacks abusing CharGen, SNMP and Quake 3.

Fearing that adversaries will soon discover other powerful amplification vectors like NTP or SSDP, we describe and evaluate countermeasures. We show that for some of the protocols, existing attack detection techniques, such as port-, content- or length-based filtering approaches will fail, as none of these attributes is characteristic in the attack traffic. In addition, we show that request rate limiting does not sufficiently protect against carefully-crafted DRDoS attacks that abuse millions of amplifiers. To foster security of future protocols and their implementations, we also discuss how protocols can be hardened and give positive examples that are presumably immune to DRDoS attacks.

The following list summarizes our contributions:

1) We discover that 14 network protocols can be abused to launch DRDoS attacks that amplify the attack traffic by the order of up to three magnitudes. We found thousands, and for six protocols even millions of amplifiers, allowing for highly distributed attacks.

2) We propose methods to detect real-world DRDoS attacks, such as bait services that are monitored for abuse and using traffic analysis to identify DRDoS victims and amplifiers. We analyze more than 130 real-world attacks, showing that DRDoS attacks bother network operators on a daily basis.

3) We describe and evaluate DRDoS countermeasures and discuss methods (not) to use to harden network protocols and vulnerable implementations.

The remainder of this paper is structured as follows. In Section II, we describe our threat model. In Section III, we show popular network protocols that are vulnerable to amplification. We search for real-world attacks abusing these protocols in Section IV. We evaluate countermeasues in Section V, discuss our findings and future work in Section VII and conclude our work in Section VIII.

II. Threat Model

Our threat model are *distributed* and *reflective* denial-of-service (DRDoS) attacks, in which an attacker A aims to consume all available bandwidth of a victim V. *Reflective* means that A does not directly send traffic to V, but instead uses systems that reflect the attack traffic to V (so called *amplifiers*). *Distributed* accounts for the fact that A abuses thousands of amplifiers and V thus faces thousands of attack sources. The victim V is any Internet-connected host (i.e., server or client) with a single uplink, usually (but not necessarily) identified by a single IPv4 address.

A controls an Internet gateway that can send IP packets with spoofed source addresses. This assumption is reasonable despite the fact that IP spoofing is considered a bad habit and is discouraged [21]. As of July 2013, the Spoofer Project listed that 25% of the Autonomous Systems world-wide allow IP spoofing [1]. Our threat model becomes more severe the higher the bandwidth available to A, but we do not make assumptions about the bandwidth an attacker can use.

A further knows at least one UDP-based protocol P for which he can craft requests that a server (a potential amplifier) for P will answer. We assume that A can obtain a set of amplifiers (M_P) that respond to valid requests of protocol P. As we will show, obtaining M_P is typically not a high burden to attackers, as attackers can easily understand most network protocols by reading public documentation or open source code. In this work, we focus on protocols for which the responses are larger than the requests. Attacks become worse the higher this imbalance is, as A can then actually amplify (and not only reflect) its traffic.

We further restrict A such that he cannot control or configure the amplifiers in any way, i.e., he can only use services offered to anybody. To be specific, for the P2P botnets we analyze, A can *not* command the bots.

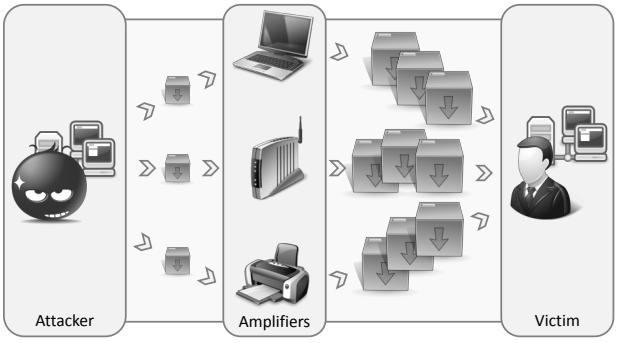

Fig. 1: Our threat model: An attacker sends requests to amplifiers with the victim's IP address as IP packet source. In turn, the amplifiers send (potentially multiple) large responses to the victim.

Figure 1 illustrates our threat model. A sends small requests to three heterogeneous amplifiers that all run services that are vulnerable to an amplification attack. A specifies V's IP address as source for the request sent to M_P, causing the amplifiers to send their responses to V — although V never asked for it. In case of amplification vulnerabilities in P, M_P typically sends responses which are significantly larger than the requests, causing bandwidth congestion at V.

Other types of DDoS attacks are outside the scope of this work, such as exhausting TCP connections or slowing down application-layer services. Note that we focus on IPv4 only, but our results should be similar in IPv6 networks.

To summarize, our threat model represents a typical setting of underground DDoS services. In fact, the most severe DRDoS attacks so far followed exactly this setting [16, 15, 36]. In this work, we seek to understand if we can expect similar or even worse attacks in the future.

III. Amplification Vulnerabilities

In DRDoS attacks, two main factors determine the impact of an attack. First, attackers strive to abuse network protocols that not only reflect traffic, but that also amplify the attack bandwidth. Second, the number of amplifiers and their aggregated attack bandwidth constrains the bandwidth of an DRDoS attack. In this section, we show which popular network protocols are attractive to attackers in this regard.

A. Protocol Overview

One of the key characteristics of DRDoS attacks is that attackers abuse network protocols in which i) small requests lead to relatively large responses and ii) where reflection of traffic with spoofed IP source addresses is possible due to the lack of a proper handshake. As such, we exclude all TCP-based protocols from our analysis, as IP address spoofing is restricted to the start of the TCP handshake. Although the TCP handshake fulfils the reflection criterion, it does not allow for easy amplification (since a TCP ACK is not larger than a TCP SYN packet). However, we show that 14 popular UDP-based network protocols are suitable candidates for DRDoS attacks.

Cat	Protocol	Port(s)	Description
Network Svc	SNMP v2	161	Monitoring network-attached devices
	NTP	123	Time synchronization
	DNS	53	(Primarily) Domain name resolution
	NetBios	137	Name service protocol of NetBios API
	SSDP	1900	Discovery of UPnP-enabled hosts
Leg.	CharGen	19	Legacy character generation protocol
	QOTD	17	Legacy "Quote-of-the-day" protocol
P2P	BitTorrent	any	BitTorrent's Kademlia DHT impl.
	Kad	any	eMule's Kademlia DHT impl.
Gam	Quake 3	27960	Games using the Quake 3 engine
	Steam	27015	Games using the Steam protocol
Bots	ZAv2	164XY	P2P-based rootkit
	Sality	any	P2P-based malware dropper
	Gameover	any	P2P-based banking trojan

TABLE I: Overview of the analyzed network protocols grouped into five categories: network services, legacy protocols, P2P file sharing, multiplayer games and P2P-based botnets. The *Port(s)* column describes the typical UDP listening port for the service or specifies *any* if ports are chosen randomly.

Table I categorizes the 14 protocols that we analyzed into five groups. The first group consists of popular protocols used for network services, such as time synchronization, host discovery or device monitoring. We added two legacy protocols, *Character Generation* (CharGen, RFC 864) and Quote of the Day (QOTD, RFC 865), to the second group. The third group comprises two protocols of P2P-based file sharing networks. In group four, we added two game server engines that are used for various game implementations. In the last group, we explore how P2P-based botnets [26] can be abused for DRDoS attacks, i.e., even for somebody not being the botmaster. This non-exhaustive list represents the protocols that we found are most relevant for our analysis. We excluded a few protocols from our analysis for which we did not find amplification vectors, such as the ECHO and ISAKMP protocols – these protocols do, however, allow non-amplified traffic reflection attacks.

B. Amplifier Enumeration

In this section, we enumerate the amplifiers per protocol, i.e., the set of public hosts that may be abused for amplification attacks. In case of DNS, dedicated projects already track the number of open resolvers that could be used to launch DRDoS attacks using DNS. However, for the majority of the other protocols, the number of potential amplifiers is unknown. The more amplifiers are available, the harder it becomes to identify or alert attack sources and the higher is the maximum attack bandwidth. We use three techniques to enumerate the amplifiers for the 14 analyzed protocols: scanning, crawling and querying master servers.

For network services, we assume that they run on their standardized UDP port and we scan the Internet for available amplifiers. To save resources and limit the noise of our scans, we scan a random sample of 10^6 IP addresses out of the set of all 2.6 billion advertized IPv4 addresses. We then extrapolate the total number of amplifiers from the number of amplifiers found during our partial scan. In our experiments, we used a synchronous 150 MBit/s Internet connection for scanning, and each scan finished in 65 seconds on average. Implementing an optimized and complete scanner is out of the scope of this work. As shown by Durumeric et al. [8], an attacker could speed up this process and complete a $/0$ IPv4 scan for one protocol in less than two hours by using a 1 Gb/s uplink and a more efficient scanner implementation. Similarly, Dainotti et al. observed botnets that complete $/0$ stealthy scans [6]. This shows that, in practice, scanning is no burden to attackers.

In the case of P2P protocols (both malicious and benign networks), we enumerate the amplifiers by an iterative search through peer list exchanges, a process often referred to as *crawling*. Crawling requires detailed knowledge about the P2P protocols and message structures. If available, we studied the source code of popular protocol implementations (e.g., eMule for Kad or libtorrent for BitTorrent), or manually reverse engineered protocol implementations otherwise (e.g., for the P2P-based bots). We obtained an initial set of ten bootstrapping peers from public resources or the protocol implementations. We then iteratively query the set of known peers to retrieve their neighboring peers. Such peer list exchanges are an inherent protocol feature of all P2P networks. Note that crawling typically only identifies Internet-facing peers, which is exactly the set of amplifiers we are interested in. That is, we include only peers that at least once responded to our requests, by which we exclude all peers behind a firewall or NAT gateway.

In order to cope with the effects of IP address churn, we terminated the crawling process after one hour. For a detailed description of our crawler implementation we refer to our crawling scheme described in our work on measuring the size of P2P-based botnets [26]. Given that we terminated our crawls after one our, we hypothesize that our P2P crawling results represent lower bounds for the actual number of amplifiers. This

would follow our detailed analysis of P2P botnet crawls [26]. We confirmed this hypothesis by observing that our crawler was still learning new peers (identified by previously unseen identifiers, if available) when terminating the process.

Lastly, for the game server protocols, we leverage the fact that game servers register at a master server, which we query to obtain a list of available game servers. Although typical, registration at the master server is not mandatory, i.e., the number of actual game servers is probably slightly higher than the set of servers obtained from the master server. We validated the retrieved server lists by cross-checking them with the set of servers displayed in-game and with online game server browsers. We also included game servers that are password-protected, as password protection in this case is irrelevant for the scenario that we will describe.

In the specific case of DNS, we measure two types of amplifiers. First, we scan the Internet for open resolvers, i.e., public resolvers that serve recursive name lookups to any client. Again, as this is not important for our threat model, we also include resolvers that returned incorrect resource records [27]. Second, we give lower bounds for the number of authoritative name servers. We cannot scan for authoritative name servers, though, as we do not know the zones a server is authoritative for. However, an attacker can crawl the Internet (e.g., web sites) for domains and then determine the authoritative name servers for these domains. Instead of crawling ourselves, we use the public dataset of the Common Crawl project[1]. We then recursively resolve each domain name and track all authoritative name servers.

We performed the measurements during weekdays between 1pm and 3pm GMT. Table II shows our results per protocol. We rounded the numbers for the scanning measurements to indicate that accuracy is lost by extrapolating the scanning results. While open DNS resolvers rank highest, also SSDP, NTP, NetBios, SNMP and BitTorrent reveal millions of potential amplifiers. Of the 255,819 authoritative DNS name servers, we found that at least 1404 deploy DNSSEC. The two legacy protocols, CharGen and QOTD, offer significantly fewer amplifiers. For the P2P networks, we ignore all peers that cannot be abused as amplifiers, e.g., because they are behind a NAT gateway or firewalled, and crawled for an hour only. The numbers are thus a subset of all peers in a network, which also explains the difference to previous measurements on P2P botnet sizes [26]. Still, the P2P networks span thousands, and BitTorrent even millions of potential amplifiers.

Lastly, we measured how long it takes for an attacker to obtain 1000 and 100,000 amplifiers per protocol, respectively. For scanning, we assume that a $/0$ scan for a UDP-based protocol finishes in two hours [8] and calculate the time to find a subset of amplifiers accordingly. For the P2P networks we measure when – relative to the time when starting a crawl – a peer responds to us the first time.

Table II shows that the time needed to acquire a reasonable number of amplifiers is negligible in most protocols. For example, it takes about a minute to acquire 100,000 BitTorrent amplifiers, and scanning for the same number of hosts completes in less than four minutes for SNMP, NTP, DNS_{OR} or SSDP. Only for protocols with fewer amplifiers, such as QOTD or CharGen, even finding a small subset of amplifiers takes several minutes. Moreover, for DNS_{NS}, crawling the web for domains (a step we omitted) and recursively resolving the domains takes hours. However, overall, this shows that attackers can assemble a set of recent amplifiers as a preparation for a DRDoS attack in no time.

[1] see http://commoncrawl.org/ – dataset as of February 2012

Protocol	Amplifiers	Tech.	t_{1k}	t_{100k}
SNMP v2	4,832,000	Scan	1.5s	148.9s
NTP	1,451,000	Scan	2.0s	195.1s
DNS_{NS}	255,819	Crawl	35.3s	3530.0s
DNS_{OR}	7,782,000	Scan	0.9s	92.5s
NetBios	2,108,000	Scan	3.4s	341.5s
SSDP	3,704,000	Scan	1.9s	193.5s
CharGen	89,000	Scan	80.6s	n/a
QOTD	32,000	Scan	228.2s	n/a
BitTorrent	5,066,635	Crawl	0.9s	63.6s
Kad	232,012	Crawl	0.9s	108.0s
Quake 3	1,059	Master	0.6s	n/a
Steam	167,886	Master	1.3s	137.1s
ZAv2	27,939	Crawl	1.5s	n/a
Sality	12,714	Crawl	4.7s	n/a
Gameover	2,023	Crawl	168.5s	n/a

TABLE II: Number of amplifiers per protocol, the technique we used to obtain the amplifiers, and the time it took to identify 1000 (t_{1k}) and 100,000 (t_{100k}) amplifiers, respectively.

C. Amplification Factors

In this section, we evaluate the potential amplification factors of the 14 protocols in Table I. In a first step, we analyzed each of the protocols to find potential vectors of amplification. For the network service protocols, we studied the respective RFC documents and popular open source implementations. We dissected the two P2P file sharing protocols by examining several open source client implementations, such as eMule, libtorrent or Vuze. For the game servers, we found possible attack vectors in the public API documentation provided by the game vendors. Lastly, we reverse-engineered the network protocols of the three bot binaries to understand the encryption schemes and different request types of the proprietary P2P protocols.

As a measure for amplification, we define the *bandwidth amplification factor* (BAF) as the bandwidth multiplier in terms of number of UDP payload bytes that an amplifier sends to answer a request, compared to the number of UDP payload bytes of the request, i.e.:

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

We chose not to include Ethernet, IP, or UDP headers in this calculation so that our measurements remain valid even if the upper protocol layers change in the future, such as during the migration from IPv4 to IPv6.

In addition, we measure the *packet amplification factor* (PAF) as the packet multiplier in terms of number of IP packets that an amplifier sends to answer a request. More formally, we define the PAF as:

$$PAF = \frac{\textit{number of packets amplifier to victim}}{\textit{number of packets attacker to amplifier}} \quad (2)$$

Protocol	BAF all	BAF 50%	BAF 10%	PAF all	Scenario
SNMP v2	6.3	8.6	11.3	1.00	*GetBulk* request
NTP	556.9	1083.2	4670.0	3.84	Request client statistics
DNS_{NS}	54.6	76.7	98.3	2.08	ANY lookup at author. NS
DNS_{OR}	28.7	41.2	64.1	1.32	ANY lookup at open resolv.
NetBios	3.8	4.5	4.9	1.00	Name resolution
SSDP	30.8	40.4	75.9	9.92	*SEARCH* request
CharGen	358.8	n/a	n/a	1.00	Character generation request
QOTD	140.3	n/a	n/a	1.00	Quote request
BitTorrent	3.8	5.3	10.3	1.58	File search
Kad	16.3	21.5	22.7	1.00	Peer list exchange
Quake 3	63.9	74.9	82.8	1.01	Server info exchange
Steam	5.5	6.9	14.7	1.12	Server info exchange
ZAv2	36.0	36.6	41.1	1.02	Peer list and cmd exchange
Sality	37.3	37.9	38.4	1.00	URL list exchange
Gameover	45.4	45.9	46.2	5.39	Peer and proxy exchange

TABLE III: Bandwidth amplifier factors per protocols. all shows the average BAF of all amplifiers, 50% and 10% show the average BAF when using the worst 50% or 10% of the amplifiers, respectively.

For each protocol, we aim to identify the request-response scenario with the highest bandwidth amplification factor. We then send the corresponding request to all amplifiers that we identify in Section III-B and measure the average response size. While the BAF mainly depends on the protocol, it also depends on the specific amplifier instance and its protocol implementation. In fact, an attacker can first measure the BAF per amplifier and – in the actual attack – use only the subset of amplifiers with the highest BAFs. We thus also included two further BAF measures that show the BAF if the attacker focuses on abusing the most severe 50% or 10% of the amplifiers, respectively. Table III summarizes our results. The PAF in Table III shows the PAF that we measured when using all (and not a subset of) amplifiers.

In the following, we will describe the maximum amplification scenarios and our measurement results per protocol:

1) SNMP v2: We found that SNMP v2[2] supports the *GetBulk* operation, in which a device returns a list of SNMP identifiers that can be monitored. In the legitimate use case, this request can be used to iterate all monitoring values. An attacker can abuse this feature to amplify traffic by factor 6.3. The exact response size is determined by the number and length of identifiers in the returned item list. If an attacker abuses only the 10% subset of the amplifiers that reply with the largest payload, the BAF increases to 11.3.

2) NTP: We found that NTP servers support the `monlist` request, which most NTP server implementations accept in its short form of only 8 bytes. Upon receiving a `monlist` request, an NTP server shares its recent clients in up to 100 UDP datagrams with 440 bytes payload each. One response datagram specifies statistics for NTP clients (such as the client's IP address, its NTP version and the number of requests) who contacted this NTP server – a useful debugging feature in the legitimate use case. The total response length depends on the number client statistics a server shares upon request. On average, `monlist` requests amplify the request traffic by factor 556.9–4670.0, the highest BAF in our measurements.

Next to `monlist`, NTP servers support further features that may be abused for attacks with significantly lower amplification rates. Preliminary research results show that these features do, however, still show severe amplification rates and are supported by a larger set of servers. As of writing this, we are still investigating the dimensions of and appropriate countermeasures against these problems with NTP server developers and CERTs. We leave a complete analysis of these further NTP features open to future work.

3) DNS: Name lookups (e.g., `A` or `MX` records), the traditional use-case of DNS, have low amplification rates. Traditionally, the size of UDP replies was limited to 512 bytes and DNS switched to TCP communication for larger replies. However, many DNS servers adopted the DNS extensions (EDNS0) that allow for UDP responses of up to 4096 bytes. Attackers may abuse `ANY` request with EDNS0, for which a server returns all known DNS record types for a given domain.

We distinguish between two different attacks abusing DNS for amplification. First, in our setting DNS_{OR}, an attacker abuses open resolvers as amplifiers. Attackers can enforce high amplification rates by resolving `ANY` requests from domains that result in large responses. In fact, an attacker could even configure a domain he controls such that its authoritative name server responds with exactly 4096-byte-wide responses. As these responses will be cached by the resolvers (i.e., the amplifiers) according to the TTL, the attacker's domain would face only little load. Depending on the queried domain name length and the maximum EDNS0 size a resolver supports, this amplifies the UDP payload by factor 28.7–64.1.

Network operators have become aware of this kind of abuse and the number of open DNS resolvers is gradually decreasing. However, there is an increasing number of authoritative name servers that include larger resource records in their responses. One of the reasons is the deployment of DNSSEC [5], in which each resource record is accompanied with a typically 1024-bit-wide signature in a special `RRSIG` record. In a second attack setting, DNS_{NS}, we thus measure the amplification rate when an attacker sends `ANY` requests to authoritative name servers only. We compute the amplification factors without the requirement to configure a specialized attack domain. In our experiment, we send `ANY` requests to the authoritative name servers found in our crawl and filter on the 1404 name servers that deploy DNSSEC for at least one domain. We then observed a BAF of 54.6, which can be even as high as 98.3 for the top 10% of authoritative name servers.

Intuitively, one would expect that the amplification rate for open resolvers is higher than for authoritative name servers, as the attacker can (as we did) control and maximize the contents of the response to an `ANY` request. However, the reason why the BAF is lower for DNS_{OR} than in DNS_{NS} is that many open resolvers did not support EDNS0 and thus chopped the responses to 512 bytes. We did not observe such a behavior for most of the authoritative name servers.

4) NetBios: In NetBios, we achieve the highest amplification using a name lookup, for which a receiving Windows system responds with its current network and host name configuration. We observed an average BAF of 3.8. The response sizes are influenced by the host names and network setups of the amplifiers. When focusing on the top 10% of amplifiers the BAF is 4.9.

[2]We focus on SNMP in version 2 as it is the most popular version and supported by the majority of SNMP devices.

5) SSDP: Upon SSDP `discovery` requests, UPnP-enabled hosts respond with one reply packet per "service" they have configured. On average, we achieved a BAF of 30.8. The response size depends on the configured services and the length of the service name ("USN"). Some of the amplifiers responded with a few reply packets only, as they offer fewer services. An attacker could thus achieve a higher BAF of 75.9 when using the top-10%-amplifiers only.

6) CharGen: According to RFC 864, CharGen servers reply with random characters to incoming UDP datagrams of any length. We therefore chose to send requests of a single byte (a technique we will later also observe to be used by attackers). On average, we received 358.8 bytes from the CharGen servers, whereas the statistical mode of the response sizes was 74 bytes and a few servers replied with up to 1472 bytes. We chose not to calculate BAFs on subsets of the servers due to lack of statistical significance for the relatively small set of 34 amplifiers in our dataset[3].

7) QOTD: Similar to CharGen, also Quote of the Day servers (RFC 865) send replies to UDP datagrams of any length. The average response size (i.e., quote length) was 140.3. Again, we do not calculate the BAF for amplifier subsets due to missing statistical significance.

8) BitTorrent: In BitTorrent, we identified that hash searches – which are primarily used to find peers that serve a specific file – may result in large responses. While normal peers only return a list of neighboring peers, BitTorrent trackers also include two 256-byte-wide bloom filter arrays with BitTorrent swarm information in their reply[4]. As not all peers know and share this information, the BAF varies among amplifiers. In our experiment, we tried to maximize the number of tracker replies by searching for the most prominent 100 file hashes[5]. While the average BAF is 3.8 only, an attacker can achieve a BAF of 10.3 by abusing the top 10% of the amplifiers (typically BitTorrent trackers).

9) Kad: We use a peer list exchange message to gain a BAF between 16.3 and 22.7 in Kad. Upon a 35-byte-wide peer list request, a Kademlia client shares up to 31 peers, which consist of an ID (16 byte), IP address (4 byte) and UDP port (2 byte) plus Kademlia message header and meta data. The number of peers shared depends on the client implementation and varied between 15 and 31 peers in our crawl. The amplification is higher compared to BitTorrent due to the smaller requests.

10) Quake 3: For the Quake 3 game servers we found the highest amplification when asking a server for its current status, a 15-byte-wide request. The reply is significantly larger and includes, e.g., the detailed server configuration and a list of current players. The average BAF was 63.9, but was as high as 82.8 for the top-10% servers with many active players and complex server configurations.

11) Steam: The Steam game protocol is used by a large number of games such as Counter-Strike, Half-Life or Team Fortress. Again, we can leverage the status information request to flood a potential victim with server status replies, with similar implications as for Quake 3. The average BAF of 5.5 is lower than for Quake 3, as the status replies do not contain the list of active players, the requests are larger (25 bytes) and the server status replies are dense. When focusing on the top-10%, we achieved a BAF of 14.7.

12) ZeroAccess: The ZeroAccess P2P botnet in version 2 (ZAv2)[6] supports three message types in its P2P communication, of which the peer list and command exchange is most effective in terms of amplification. For a request of only 16 bytes, a ZAv2 bot returns 16 neighbors (8 bytes each) and information about the currently active malicious modules, including a 128-byte-wide RSA signature by the botmaster. This results in BAFs of 36.0-41.1. Usually bots are synchronized regarding the commands, i.e., we observed only a low amplification derivation between all responses.

13) Sality: Similar to ZeroAccess, also the Sality P2P bot offers three message types as part of its command-and-control communication. Sality is a malware downloader and bots can thus exchange URL lists of files that bots should install on the infected PC, including a 256-byte-wide RSA signature. We measured a BAF of 37.3–38.4. Again, most bots are synchronized regarding the current URL lists, so there is only little gain in choosing amplifier subsets.

14)Gameover: In the Gameover P2P bot, a banking trojan, we leverage the peer list and proxy list exchange mechanism to achieve highest amplification. For a 52-byte-wide UDP payload, Gameover bots first reply with a list of up to ten neighboring peers [2]. In addition, bots send four additional datagrams with a list of proxies that can be used to upload data, such as stolen online banking credentials. The average number of packets is even larger: Gameover peers may issue counter-requests to amplification victims, an artifact that increases the packet amplification above 5. In total, Gameover offers a relatively constant BAF of 46.

IV. Real-World Observations

The amplification potential identified in the previous section is attractive to attackers who follow our threat model. In this section, we analyze to what extent such attacks can be observed in the wild. First, using Netflow data from a large ISP, we search for DRDoS attack victims and for amplifiers that are abused during DRDoS attacks. Second, we search for UDP port scans in darknet traffic, which may indicate that attackers try to spy on amplifiers. Third, we publish and monitor amplification *baits* which are attractive to be abused in amplification attacks.

A. Dataset Descriptions

1) Netflow data: We obtained Netflow data from a large European ISP comprising about 1 million end users (consumer and business). The ISP hosts multiple servers that are vulnerable to amplification attacks. We manually confirmed the presence of open DNS resolvers, authoritative name servers, various game servers, NTP servers, SNMP- and SSDP-enabled hosts, four CharGen servers, dozens of clients infected with P2P-bots and plenty of P2P file sharing users. The ISP blocks

[3]Note that the number of amplifiers availablet to us is so low because we only scanned a subset of all available IPv4 addresses

[4]See BEP 33: http://www.bittorrent.org/beps/bep_0033.html

[5]We obtained this list from http://thepiratebay.sx/top/all

[6]ZeroAccess version 1 has been reported to contain significantly fewer peers [26] and we therefore did not further analyze it

IP spoofing for egress packets, but we can still use the dataset to detect incoming DRDoS attacks and abuses of amplifiers within the ISP's network(s).

The dataset spans from July 14th to July 25th 2013 (12 days). Due to the high load on the routers, the ISP sampled Netflow data and configured the routers to randomly choose 1/100th of all packets for Netflow recording. Luckily, due to the aggressive nature of DDoS attacks, such a sampling does not interfere with our goals to detect the attacks. We filter the Netflow data on the UDP ports of the protocols with a fixed server port (see Table I on page 3) to ease analysis. We will thus miss a few protocols in favor of a higher data processing performance.

2) Darknet traffic: Darknets are unused IP address ranges, i.e., networks in which no services or users are hosted and that should thus not be used in legitimate cases. Any traffic directed to the darknet can be considered as background Internet traffic, such as backscatter or scans. Darknets are thus a suitable setting to identify scanning activity [3, 40, 6]. We have access to two darknets with different granularity. Darknet $D1$ spans a $/17$ network and is monitored using Netflow. Darknet $D2$ spans a disjoint $/27$ network and we record the full network traces to allow packet inspection. We use four weeks of traffic from June/July 2013 in both darknets for further analysis.

3) Amplifier baits: Third, we published *bait* services on the Internet that are vulnerable to amplification abuse. We tried to make these baits public and attractive to attackers. For example, we hosted game servers for Quake 3 and Steam, registered them at the Master server and added bots (non-human players) to increase the amplification factor for these servers. We started a public NTP server and listed it in public NTP pools. In addition, we spawned file sharing clients for BitTorrent (`uTorrent`) and Kad (`eMule`) and let them download and seed popular files. We also have access to a real authoritative name servers that hosts a DNSSEC-enabled domain with Google's PageRank popularity of five. We implemented and launched CharGen and QOTD services. Finally, we infected each a separate Windows VM with one of the malicious P2P-based bots and limited the network connectivity to the P2P communication, mitigating all potential threats by the bot such as spam or clickfraud. We exposed these services end-June[7] and have been operating them for four consecutive weeks before analyzing the data.

We operate the baits on public IPv4 addresses to prevent side-effects from firewalls or NAT gateways. This may raise ethical concerns, as we publish services that can be abused for DDoS attacks, i.e., our baits may participate in real attacks. To mitigate this harm, we strictly monitored these systems and throttled the maximum uplink bandwidth for each of the services to 1 MBit/s. In addition, when we saw that an attack started, we immediately blocked the outgoing traffic on a per-IP address basis. This way, even if the bait services are abused for amplification attacks, the attack impact is limited and represents less bandwidth than a typical amplifier would have. Admittedly, though, we could not entirely avoid participating in actual attacks. We hope, however, that the insights that we got compensate the potential harm of our experiments.

[7]The authoritative DNS servers under analysis are already in use since 2008 and the P2P-based bots were launched in April 2013.

B. Real-world Amplification Scans

We first aim to identify UDP-based scanning activities that may indicate attackers who search for amplifiers. As we have shown in Section III, amplifiers for most protocols can be found by horizontally scanning the Internet for open services. With horizontally we mean that attackers choose one service that they want to abuse and then scan (a subset of) the IP address space for potential amplifiers for this service. Such Internet-wide scans can be identified using darknets [6, 40, 3]. We leverage our two darknet datasets, $D1$ and $D2$, to inspect what services are prominent to be scanned by potential attackers. We obtain a concise overview by aggregating the packets received in our darknets per UDP port. Detecting TCP-based scanning activity is unrelated to bandwidth amplification attacks and is therefore outside the scope of this work.

		Darknet $D1$			Darknet $D2$		
Port	Proto	Pkts	pps	Bpp	Pkts	pph	Bpp
5060	SIP	4.9M	2	412	18192	32	410
53	DNS	4.4M	1	38	12102	21	34
19	CharGen	1.7M	0	18	4552	8	1
4614	*unknown*	1.0M	0	66	n/a	n/a	n/a
1434	MSSQL	0.7M	0	375	958	1	356
3544	Teredo	0.4M	0	55	492	<1	49
137	NetBios	0.3M	0	53	34696	62	48
39455	*unknown*	0.2M	0	30	564	1	25
161	SNMP	0.1M	0	43	200	<1	87
5061	SIP	0.1M	0	410	2420	4	413

TABLE IV: Most popular UDP scans as found in the darknets. *Pkts* shows the absolute number of packets during the measured interval, *pps* and *pph* show the packet rate in seconds and hours, respectively. *Bpp* indicates the average UDP payload size in bytes.

Table IV summarizes the traffic statistics for the top-10 most prominent UDP ports in darknet $D1$. We will mainly use $D2$ to manually inspect the scans, but for completeness, we also list the numbers of $D2$ in the table. Table IV shows four protocols that we identified as vulnerable to amplification attacks: DNS, CharGen, NetBios and SNMP. We use the network traces recorded at $D2$ to manually inspect the traffic for these protocols.

From our set of vulnerable protocols, DNS is clearly most popular and even in the smaller darknet $D2$ we found 176 different hosts scanning for open resolvers. Interestingly, we identified quite a few security groups exploring open DNS resolvers, such as dnsresearch.us, openresolverproject.org, dnsscan.shadowserver.org and jupitoris.jaist.ac.jp. However, 27% of the scan sources requested recursive ANY lookups, an indication they search for open resolvers with high amplification rates. One attacker used 94 IP addresses from a $/23$ network to resolve www.google.com. Even in the small darknet $D2$, we recognized 21 different request patterns (grouping based on requested type, domain and EDNS options), showing that various parties are interested in finding open resolvers.

For CharGen, all hosts sent one byte UDP payload to scan for servers, in all cases except one the value 0x01. This may indicate that the code used for scanning for CharGen servers is shared among attackers. Similarly, all the NetBios scans used equal name queries, except that the transaction ID was always randomized and the broadcast flag was set in a few cases – possibly an artifact of the Conficker worm [22]. For SNMP,

we saw that scanners searched both, for SNMP v1 hosts (63%) and SNMP v2 hosts (37%), using a mixture of bulk- and non-bulk requests all in the "public" community[8]. NTP and SSDP were not present in the top-20 list of $D1$. We also did not find NTP `monlist` requests in $D2$, showing (yet) little interest by attackers in these protocols.

The darknets reveal also scans for protocols other than the ones we discuss in this paper. For example, most prominent, attackers scan for SIP-enabled devices (UDP port 5060/5061). However, the high number of average bytes per packet (>400) suggests that these scans are not related to amplification attacks. Likewise, the scans for MSSQL servers rather indicate other scanning motivations, e.g., identifying devices vulnerable to exploitation. The low packet sizes for scans for Teredo tunnels and two unknown services (ports 4614/39455) may suggest attackers search for devices vulnerable for amplification attacks. However, manual payload inspection did not reveal the intention of these scans.

C. Real-world DRDoS Victims

In this section, we aim to detect DRDoS victims in an ISP's network. We follow the intuition that DRDoS victims will receive large amounts of traffic from amplifiers, while they have never requested data from the amplifiers.

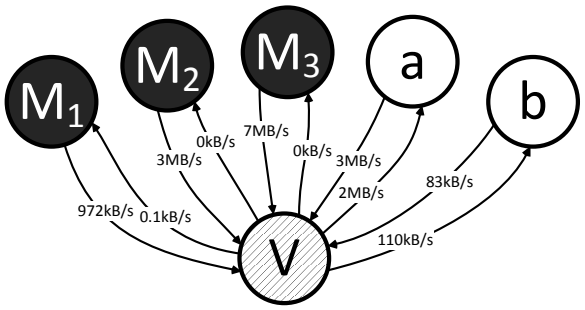

Fig. 2: A DRDoS attack against a host V with three attacking amplifiers (M_1, M_2, M_3) and two legitimate hosts (a, b). The arrows show the average bandwidth per communication stream.

Figure 2 shows a typical amplification attack scenario. In this example, three amplifiers (black nodes) are abused by an attacker to relay large amounts of traffic to the victim. Due to IP address spoofing, the attacker remains invisible to the victim. However, from the victim's perspective, the amplifiers can be detected by correlating the in- and outgoing traffic per client. For example, while amplifier M_2 sends traffic with an average bandwidth of 3 MB/s, V never sent any requests to M_2. In the normal case, such as for legitimate hosts a and b, we observe a more balanced ratio between in- and outgoing traffic. Note that a high bandwidth towards a host alone is not only characteristic for DRDoS attacks. In Figure 2, nodes V and a may, e.g., transfer large files using UDP, causing high transfer rates.

We leverage these observations to detect victims of amplification attacks using the Netflow dataset. For each client/server pair, we extract a few network characteristics by aggregating Netflow data. In this context, a *server* runs a specific service that is vulnerable to amplification attacks, thus the servers represent the set of potential amplifiers. We restrict our analysis on network protocols that use a fixed UDP port and we can thus identify servers by their (IP address, UDP port) tuple. For example, to find DNS servers, we search for hosts that send UDP packets with source port 53.[9] A *client* is any host, identified only by its IP address (ignoring client ports), communicating with a server. We consider all clients as potential DRDoS victims. In the rare case if both hosts of a flow communicate on server ports, we treat the hosts both as client and as server.

We then create a *pairflow* for each client/server pair and aggregate a few light-weight communication features:

$$pairflow := \ < C_{IP}, S_{IP}, S_{port}, B_{2s}, B_{2c}, t >$$

A client/server pair is identified by the client IP address (C_{IP}), the server IP address (S_{IP}) and the server's UDP port (S_{port}). We generate multiple pairs per client/server if a server runs multiple services. For each pairflow, we then extract the number of UDP payload bytes (i.e., in accordance with our definition of BAF in Section III) per communication direction, whereas B_{2s} is the number of bytes from the client to the server and B_{2c} vice versa. We focus only on UDP traffic and ignore other protocols like TCP or ICMP. Lastly, we compute the pairflow duration t, defined as the interval between the first and the last packet of the client/server pair. We can use t to compute average bandwidths of pairflows.

We compute the pairflows for each hour in our Netflow dataset. That means, for the 12 days of our measurement, we compute pairflows for 288 hours separately. This allows us to speed up computation and decrease the latency between the attack and attack detection. We then apply a heuristic to find clients that are potential DRDoS victims in that we i) filter on pairflows exceeding a traffic threshold and ii) filter on clients that received significantly more traffic than they sent to the server. For i), we discard all pairflows f where $f.B_{2c} < T_B$ and we set $T_B = 100,000$ bytes in our evaluation. We use this optional pre-processing filter to speed up the process of matching client and server pairs. Setting lower traffic volume thresholds also helps to discard backscatter or scanning-related packets [19, 3]. In addition, discarding less prominent flows helps to cope with packet sampling, as bi-directional flows with a few packets may turn into uni-directional flows in sampled Netflow data. Finally, we also discard all pairflows with servers that are within the network of the ISP, as we do not consider threats from within the network.

For ii), for all remaining pairflows, we compute a ratio between sent and received bytes, i.e., $r_f = \frac{f.B_{2s}}{f.B_{2c}}$. If a client never sends requests to the server, i.e. $B_{2s} = 0$, we set $r_f = \infty$. Most pairflows between the amplifiers and the victim during a DRDoS attack have $r = \infty$, as it is unlikely that the victim requests data from the amplifiers[10]. We mark

[8]In SNMP, the *community* string is used for authentication purposes, the default setting of "public" for most devices is therefore wide-spread.

[9]Note that the UDP server port alone is no guarantee for that a server runs a specific protocol. However, in this context it is only important that an amplifier always uses a fixed port, and the port number itself only gives a quick and unreliable indication about the server type.

[10]There are legitimate cases in which victims request data from amplifiers (e.g., for normal DNS resolution). However, this affects only a few pairflows and we would still detect suspicious pairflows from all other amplifiers.

pairflows as suspicious if the BAF exceeds a threshold T_r, i.e., if $r_f > T_r$. In our evaluation, we chose a conservative setting of $T_r = 1000$, i.e., raise suspicion for all servers (potential amplifiers) that send more than 1000x traffic volume to the client (potential victim) than they received from it.

Results: Table V lists the 15 DRDoS attacks against subscribers of the ISP that our analysis on the Netflow dataset revealed. All times listed in the table are GMT. We assigned a unique label (A–I) to the victims and hide IP addresses for privacy reasons. As expected, the majority of the attacks involved DNS amplification. The largest attack spanned 711 MBit/s from 330 amplifiers and lasted for three hours. The shortest attack lasted for 39 sec. with a bandwidth of 212 MBit/s. Interestingly, attackers already abuse further protocols. We found four CharGen-based attacks that involved up to 3065 amplifiers. Against victims C and G, we observed attacks abusing both, DNS and CharGen simultaneously. Victim F was first targeted with a low-volume CharGen-based attack, until the attackers switched to DNS and achieved almost 1 Gb/s attack capacity.

Attack time span	Port	V	$\|M\|$	Volume	BW
07/14 00:41:18 – 02:48:46	53	A	2631	28238	29
07/17 17:49:25 – 18:04:36	53	B	8501	15300	134
07/18 01:41:57 – 01:56:08	53	C	255	13260	124
07/18 01:42:00 – 01:56:12	19	C	43	7306	68
07/19 22:21:44 – 23:10:32	53	D	7907	7567	20
07/20 12:32:33 – 21:21:41	53	D	16339	36314	9
07/20 18:18:20 – 18:24:44	53	E	367	4227	88
07/21 14:27:45 – 14:43:55	19	F	3065	2619	12
07/21 14:42:13 – 17:43:50	53	F	330	968873	711
07/21 18:03:31 – 19:01:40	19	G	814	25296	58
07/21 18:04:14 – 18:12:47	53	G	453	11503	179
07/22 12:40:35 – 12:49:54	19	H	1151	3841	54
07/22 13:02:52 – 13:08:06	53	D	10573	8317	211
07/22 15:38:13 – 15:38:52	53	I	193	1034	212
07/23 12:59:42 – 20:38:42	53	D	14187	16662	4

TABLE V: List of DDoS victims found in the Netflow dataset. V denotes the victim's identifier, $|M|$ the number of amplifiers, *Volume* the total attack volume measured in MB, and *BW* the average attack bandwidth in *Mbit/s*.

We verified our findings by discussing them with the ISP's CERT. All the DNS-based attack victims were known to the CERT due to basic alerting systems in their networks and the attacks could be confirmed. Three of the CharGen-based attacks could be confirmed by correlating the victim's IP address with the target of preceding DNS-based attacks. The fourth CharGen-based attack, although relatively small in bandwidth, severely interfered with a DSL-connected customer. The CERT was not aware of any DRDoS attack that we did not detect, indicating a low false negative rate. However, the CERT had been affected by SNMP-based amplification attacks outside of the timespan that we monitored. This shows that attackers already abuse some of the protocols which we found to be prone to DRDoS abuse.

D. Real-world Amplifier Abuse

Next, we aim to identify legitimate services that are abused as amplifiers during a DRDoS attack. Our goal is thus not to identify the DRDoS victim (as in Section IV-C), but to detect the amplification abuse itself as a first step towards service hardening and egress filtering.

Detecting the abuse of amplifiers is significantly more difficult than identifying DRDoS victims. The ratio between in- and outgoing bytes of victims was quite distinctive in the previous context. With amplifiers, a few complications are added to the detection. First, amplifiers act as servers and are thus also used by many other, legitimate clients. Legitimate clients may have the normal demand for high-bandwidth communication, thus (again) the bandwidth alone is not a good indicator for abuse. Second, as opposed to DRDoS victims, amplifiers always have both, incoming and outgoing traffic within a client/server pair.

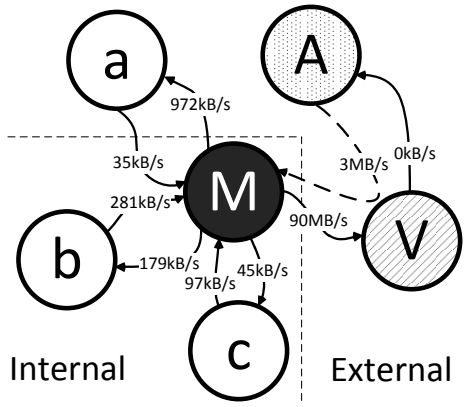

Fig. 3: An scenario where A abuses M as amplifier to attack V, while M also has two legitimate clients in the internal network (b, c) and one external client (a).

Figure 3 illustrates a setting in which an amplifier M is used by three legitimate clients, two of which are internal (b, c) and one outside the ISP's network (a). An attacker A abuses the amplifier to reflect traffic to an external victim V. To detect this abuse in a similar way than we detected the DRDoS victims, we have to relax the constraint of the expected ratio between in- and outgoing bytes. From the ISP's perspective, V and M have a relatively normal conversation, as the ISP cannot detect that the attacker spoofs V's IP address. What remains remarkable is a high amplification ratio of factor 30 between in- and outgoing bytes between M and V. However, this alone is not sufficiently descriptive, as legitimate clients may also show similar behavior. For example, the legitimate client b may request DNSSEC records from M, resulting in similar amplification rates.

Nevertheless, we can re-use the technique presented in Section IV-C, but with different filtering thresholds. Based on our insights on average amplification rates from Section III, we chose to flag pairflows with (more than) a fivefold BAF, i.e., we set $T_r = 5.0$. To focus on aggressive attacks and limit the number of false positives, we set the threshold for the minimum number of bytes sent by a potential amplifier to 10 MB, i.e., $T_B = 10{,}000{,}000$. We further aim to detect amplifiers within the ISP's network. We thus discard all pairflows with external servers (potential amplifiers), i.e., servers that are outside of the authority of the ISP (and its customers). In favor of fewer alerts for manual inspection, we also discard pairflows with an average bandwidth of less than 10 Kbit/s. This may cause that we miss low-volume attacks, but it helps to filter out the large number of legitimate client/server pairs with low-volume communication.

Results: Our method flagged 143 pairflows in the Netflow dataset as suspicious. We again discussed our insights with the CERT to evaluate our results. We spotted six open recursive DNS resolvers that were abused in 55 DRDoS attacks with BAF between 34–67. In addition, one authoritative name server was abused in three DRDoS attacks. Our methodology also flagged four closed DNS recursive resolvers as suspicious. These servers were mistakenly flagged as amplifiers as they tried to resolve domains from authoritative name servers (all of which at mail.ru) that faced packet loss and thus rarely responded to requests. In these cases, the number of packets and therefore the volume sent was much larger than the volume of the received responses.

Next to DNS, we identified four CharGen servers that were abused to attack 57 victims. The amplification factors varied per CharGen server from 318 to 1395 and was determined by the length of the response string by the server. We also found five SNMP-enabled devices that showed suspicious behavior. In two separate time spans, one 55 min, another 1:20h, these devices were abused for simultaneous low-volume attacks with 35 Kbit/s each. Lastly, we found alerts for potential abuses of three Steam game servers. Unfortunately, we could not verify these alerts, as the servers had already been taken offline when we manually investigated them.

In a different experiment, we inspected the traffic recorded at our amplifier baits. We observed one case where our CharGen server was abused as DRDoS amplifier. The attack started just 14 minutes after the server was first discovered by a host – showing how fast attackers act from identifying amplifiers to abusing them. In seven further incidents, we found that our Quake 3 game server was abused, in all cases exactly in the way we describe the attack in Section III. None of the baits in the P2P networks, neither the benign networks nor the malicious botnets, were abused in DRDoS attacks. We had to ignore eight alerts that were caused by UDP-based and session-aware file downloads, though, which legitimately have high amplification factors.

Overall, we have identified dozens of real-world amplification DRDoS attacks, showing that amplification vulnerabilities are indeed a problem. Attackers are already exploiting the amplification potential of game server protocols, CharGen, DNS and SNMP, and will likely explore further protocol vulnerabilities in the future.

V. Countermeasures

We have shown that DRDoS attacks pose a problem to network administrators, ISPs or even Internet Exchange Points, which is also documented by past attacks [16, 15, 36]. In this section, we therefore discuss proactive and reactive countermeasures against DRDoS attacks.

A. Preventing IP Address Spoofing

The root cause of amplification attacks is that an attacker can force amplifiers to reflect responses to victims by spoofing the source IP address. If spoofing was not possible, our threat model and hence all attacks would be mitigated. Lots of effort has thus been spent on disabling IP source address spoofing. For example, Ferguson and Senie suggest to drop all packets that do not have IP source addresses an exit router is responsible for [21]. Many providers acted and prevent spoofing nowadays so that attackers cannot abuse their networks for amplification attacks.

Unfortunately, still a significant number of providers allow IP address spoofing. The Spoofer Project, to our knowledge the best public resource for measures of such kind, stated in August 2013 that 24.6% of the Autonomous Systems fully allow and further 13.9% at least partially allow IP address spoofing [1]. Geographical tests show that the networks that support spoofing are distributed worldwide, i.e., we do not face a regional problem. While we advocate to ban IP spoofing, it remains unclear if at some point all networks will mitigate IP address spoofing.

B. Protocol Hardening

Despite IP address spoofing, we found that many protocols were designed without considering the security implications of DRDoS attacks. In this section, we will discuss complementary ways to harden protocols against amplification abuse.

1) Session Handling: One of the core issues we identified is that amplification attacks are possible if UDP-based protocols do not require sessions. For example, the three-way handshake in TCP would not complete for spoofed IP addresses and thus effectively mitigates our threat model. A possible solution for the vulnerable UDP-based protocols is thus to include similar session handling.

In fact, a few more recent protocols already include session handling. For example, the QUIC protocol, an UDP-based version of Google's SPDY protocol, assigns a source address token to the client. The token is an authenticated-encryption block that contains the client's IP address and a timestamp. Receipt and successful retransmission of the token by the client is taken as proof of ownership of the IP. Similarly, in some BitTorrent tracker implementations, the servers compute a "connection ID" and send it to the client. Only if the client retransmit this ID in future requests the tracker will respond. In Steam, clients have to request a 4-byte-wide challenge before they can request more elaborate server information, such as the list of players. In DTLS, a UDP-based variant of TLS, the server can issue stateless cookies to a client, which a client must include in follow-up communication. The server proceeds communication only if it can verify the cookie.

Session handling prevents amplification attacks only if all request types vulnerable to amplification demand prior session instantiation. For example, in the Steam protocol, we found a BAF of 14.7 for the request types that do *not* require prior session agreement. The other downside of sessions is that they decrease the efficiency of protocols. First, establishing a session may add latency to the initial communication. Second, the client needs to prove its session within each request, adding extra bytes to each request.

Lastly, it is tricky – if not impossible – to add session handling to existing protocols without breaking compatibility with legacy clients. Session handling typically needs to be included during a protocol's design phase and before the protocol is deployed. A possible solution for protocols that use UDP *and* TCP (such as DNS) is switching to TCP mode if responses exceed a size threshold.

2) Request/Response Symmetry: An alternative strategy to reduce the amplification rate is demanding that requests are similarly large as the expected responses. A server that receives a request that is "too small" will not, or only partially, respond to the request, without the need of any session handling. Protocols hardened in such a way are not vulnerable for amplification attacks, which causes that many of the existing protocols cannot be abused by attackers. The downside is that the efficiency of the protocol drops and the clients/servers face higher loads even in the benign use case.

3) Rate Limiting: Another strategy to harden a protocol is to limit the number of requests a client may issue. In fact, a few protocol implementations already deploy rate limiting. For example, a Reponse Rate Limiting feature is currently integrated into popular DNS name servers [38]. Administrators can configure DNS servers such that they limit the number of responses per subnet in a given time interval, falling back to truncated messages if the limit is exceeded. Recent Quake 3 server implementations only respond with a single status report per requesting IP address and second. Kad implementations like `eMule` count the client requests and blacklist peers that send too many requests. And even in malicious networks like the Gameover botnet rate limiting is in place – IP addresses are permanently blacklisted if they issue more than ten requests per minute [2].

Rate limiting helps to protect against excessive abuse of a single amplifier. However, an attacker may choose to abuse an amplifier at a low request rate, such as one request per second. A single amplifier does not exhibit large amounts of traffic in such an attack. Combining millions of amplifiers results in high accumulated attack bandwidths, though. We therefore seek to understand how powerful amplification attacks can be even in presence of rate limiting.

Protocol	Amplifiers	resplen	BW in Gb/interval /32	/24
SNMP v2	4,832,000	257.5	9.8	2546.7
NTP	1,451,000	4454.8	51.7	13,240.0
DNS_{NS}	1,404	1178.2	0.0	3.4
DNS_{OR}	7,782,000	2238.6	4.6	1,172.8
NetBios	2,108,000	191.3	3.2	826.1
SSDP	3,704,000	2917.2	86.8	22,225.1
CharGen	89,000	358.8	0.3	65.7
QOTD	32,000	140.3	0.0	9.1
BitTorrent	5,066,635	360.8	14.6	3,743.8
Kad	232,012	543.8	1.0	258.4
Quake 3	1,059	831.2	0.0	1.8
Steam	167,886	136.7	0.2	47.0
ZAv2	27,939	575.4	0.1	32.9
Sality	12,714	522.8	0.1	13.6
Gameover	2,023	1999.2	0.0	8.3

TABLE VI: Aggregated DRDoS bandwidth per protocol for a victim ($/32$) and a victim's network ($/24$) if rate limiting is deployed.

To analyze the effects of rate limiting, we assume for a gedanken experiment that all protocols rate limited the requests, e.g., to one request per source IP address and second. We can then multiply the number of amplifiers with the length of the response to calculate the aggregated attack potential. Table VI summarizes the total bandwidth an attacker could still gain in such a setting (column $/32$). For protocols with small numbers of amplifiers, such as in DNS_{NS}, rate limiting can indeed limit the harm of attacks to less than 100 Mbit/s. However, for six other services, the aggregated attack bandwidth is still in the Gb/s range.

Worse, attackers can evade host-based rate limiting to some extent by spoofing multiple IP addresses from a victim's network. This is possible, for example, if adversaries attack a network instead of a single host. We thus also compute the bandwidths a victim with an attack surface of a $/24$ network would face. With as few as 256 hosts in a network, rate limiting fails to prevent larger attacks and allows attacks from 32.9 Gb/s to 22.2 Tb/s for nine protocols. We thus advocate to perform rate limiting on a per-subnet basis instead of a per-host basis.

Finally, badly-designed rate limiting implementations may allow for dangerous DoS attacks against the protocol itself. For example, an attacker could aim to exhaust the rate of a DNS resolver at authoritative name servers to prevent it from resolving domains. Luckily, the DNS-based rate limiting initiative has countered this by falling back to truncating or TCP. Similarly, an attacker can prevent a player from retrieving game server status information by exhausting the allowed rate for the player's IP address. We leave an evaluation of similar attacks open to future work.

C. Secure Service Configuration

Many protocols can be hardened against amplification by fixing weak parts in the service configurations. For example, configuring a DNS name server that offers recursive name resolution to the public is considered bad practice. To secure their servers, administrators can restrict the authorized clients to the users in their network and discard requests from all other address ranges. Moreover, authenticating clients would help to protect other services. For example, most of the SNMP-enabled devices are exposed to the Internet and use the default community without password protection – configuring a password mitigates abuse.

The reasons for insecure server configurations are manifold. A major issue are services that are simply enabled with insecure default configurations. For example, vendors often ship devices with configurations that are vulnerable to amplification attacks, such as network-enabled printers or consumer routers. In a few cases, a poorly designed protocol (or one of the protocol's features) allows attacks and changing configurations does not help – such as in the cases of the legacy protocols QOTD and CharGen.

D. Packet-based Filtering

Reactive countermeasures will help as a last resort against DRDoS attacks. Defenders, such as upstream providers, can typically deploy packet-based filtering techniques to block attack traffic. We asked a CERT about the features their anti-DDoS appliances offer for packet-based filtering. We understood that they typically use four packet-based filtering techniques to mitigate DRDoS attacks: filters by (i) IP addresses, (ii) UDP and TCP ports, (iii) packet lengths, or (iv) by payload string matching.

We seek to understand to what extent DRDoS attacks abusing the 14 protocols can be detected by such packet-based filters. The distributed nature of DRDoS attacks evades IP address filters. We thus analyze if the protocols have

characteristic behavior that matches any of the latter three packet-based filtering features. We proceed as follows for each protocol. For (ii), we measure the number of UDP ports amplifiers use to reply to our requests, and compute what ratio of the amplifiers answered with the most popular UDP port (the statistical mode). For (iii), we proceed likewise for the response lengths. For (iv), we search for static substrings in all responses which can be used for payload matching. We explicitly do not analyze false positives of packet-based filtering approaches. These approaches are crude (but helpful) methods to separate attacking protocols from other protocols and are often the only alternative to null-routing DRDoS victims.

Table VII summarizes our results. For the second and third major column, we list first the number of unique ports/lengths and then the ratio of ports/lengths matching the statistical mode.[11] The fourth major column shows the number of bytes we found to be static in all replies (at a fixed offset). The '+' indicates that further payload bytes are fixed, but at varying packet offsets. For example, we observed NetBios responses from six different source ports that all share at least seven payload bytes – a vast majority (97.9%) of these packets are sent using the standard source port, though.

Protocol	(ii) UDP ports		(iii) Resp len		(iv) PL	Detection Port	Detection len	Detection PL
SNMP	1	100.0%	239	14.9%	+9B	✓		✓
NTP	1	100.0%	90	26.1%	>100B	✓		✓
DNS_{NS}	—	—	875	2.1%	+7B			✓
DNS_{OR}	> 1000	41.3%	70	24.7%	+7B			✓
NetBios	6	97.9%	21	29.1%	+55B	✓		✓
SSDP	1	100.0%	96	36.0%	+17B	✓		✓
CharGen	1	100.0%	5	76.5%	+36B	✓	✓	✓
QOTD	1	100.0%	10	16.7%	+1B	✓		
BitTorrent	> 1000	12.4%	128	24.1%	+12B			✓
Kad	> 1000	17.2%	54	54.8%	2B			
Quake 3	174	41.7%	462	0.8%	+19B			✓
Steam	> 1000	8.9%	856	19.9%	+8B			✓
ZAv2	84	98.6%	13	98.3%	+12B	✓	✓	✓
Sality	> 1000	2.1%	33	3.7%	none			
Gameover	> 1000	0.3%	201	3.3%	none			

TABLE VII: Packet-based filtering vectors to detect protocol characteristics. The second major column shows if UDP source ports are characteristic, the third analyzes the response length, and the fourth shows the number of static bytes for payload inspection.

Table VII shows that we can use the source port to match over 95% of the packets for seven protocols. Although not a clear attack indicator, it gives a first opportunity to filter on other attack criteria, such as content filters. Length-based filtering looks ineffective for most protocols, but matches 98% of the packets of the ZeroAccess botnet[12]. Lastly, 10 of the 14 protocols have a static substring of at least seven bytes, allowing for payload-based detection. Having said this, these packet-based detection mechanisms do not necessarily help to detection attack traffic. Instead, they can typically only pre-select candidates for attack traffic based on typical characteristics per protocol — distinguishing legitimate from malicious traffic remains tricky.

A few services stand out of the set, as they do not offer any known way for detecting attack traffic. For example, the Kad responses can only be identified by matching two bytes of payload – source ports are chosen at random and responses vary in length. In fact, an attacker abusing Kad can influence the response length by varying the number of peers he requests during a peer exchange. Worse, for two botnets, the responses are encrypted in such a way that the payloads lack characteristic substrings, source ports are chosen at random, and responses are padded with a random number of bytes. This leaves defenders little chance to identify these protocols on a per-packet basis.

[11] We accidentally did not record the source ports in the experiments for DNS_{NS} and thus had to omit their analysis.

[12] In fact, ZAv2 spans four separate botnets each running one port. In this special case, we used these four ports as a mode.

VI. Related Work

This section describes the related work in the are of DRDoS attacks. Specht/Lee [30] and Mirkovic/Reiher [18] gave a general overview and proposed a taxonomy of DDoS attacks. We will discuss more specific related research in the following subsections grouped by topic.

A. DDoS Attack Types

An alternative way to launch powerful DDoS attacks are networks of remotely-controllable bots that are abused to craft DDoS attacks. Büscher and Holz analyze DirtJumper, a botnet family with the specific task to perform DDoS attacks by abusing the Internet connection of infected PCs [4]. The DirtJumper botnet attacks at the application-level layer and does not aim to exhaust bandwidth, though. Kang et al. propose the Crossfire attack, in which bots direct low-intensity flows to a large number of publicly accessible servers [12]. These flows are concentrated on carefully chosen links such that they flood these links and disconnect target servers from the Internet. Studer and Perrig describe the Coremelt attack, in which bots send legitimate traffic to each other to flood and disable a network link between them [32]. All these attacks rely on bots, while our threat model only assumes that an attacker has any spoofing-enabled Internet uplink. Although the DRDoS attacks primarily try to congest bandwidth of a single victim, they can possibly be combined with the aforementioned techniques.

Naoumov and Ross analyze if P2P systems can be abused for DDoS attacks by aggressively advertizing a victim in the Distributed Hash Table (DHT) [20]. They manipulate the Overnet P2P network and show that a potential victim will receive undesired TCP connections and UDP packets once it is well-known among the peers. Experiments have shown that this attack adds two Mbit/s bandwidth and 350 simultaneous TCP connections to victims – clearly falling behind the abuse potential we have identified for P2P networks. Sia [29] and El Defrawy et al. [7] describe similar attacks poisoning the BitTorrent DHT with equally low DDoS attack volumes. Sun et al. found that a similar attack on Kad DHT allows an attacker for bandwidth amplification of up to factor 8 [33]. Further *stateful* attacks have been evaluated by Sun et al. [34, 35] even for gossip-based P2P networks that do not have a DHT. However, stateful attacks require peer membership poisoning, which is impractical for attackers. To the best of our knowledge, we are the first to analyze the amplification vulnerabilities in P2P protocols that can be abused for DRDoS attacks. In contrast to attacks that manipulate P2P membership information, we described *stateless* attacks. Stateless attacks can be launched and stopped without delay in bootstrapping

or shutdown, which attackers may abuse for attacks that require fast reaction times (such as extortion attacks).

Closest to our work, Paxson analyzed the possibility for DRDoS attacks in 2001 [24]. Paxson's threat model is reflection in general, including the abuse of (non- to little-amplified) protocols such as ICMP and TCP. He also mentions two UDP-based protocols, DNS and SNMP, as possible amplification vectors. We have discovered amplification vectors that evade nowadays' detection methods and discovered 12 further UDP-based protocols that are also vulnerable to amplification attacks. In addition, we demonstrate and measure the threat of amplification attacks, for which only anecdotal evidence existed until now.

B. DDoS Detection and Mitigation

Another field of research investigated techniques how to detect and mitigate DDoS attack. Ioannidis and Bellovin proposed Pushback [10], in which upstream routers collaborate to drop undesired packets. Wang et al. propose to counter bandwidth-congestion attacks by forcing senders to solve small computational puzzles [39]. Sekar et al. propose LADS, a Netflow and SNMP-driven system to detect anomalies of DDoS traffic [28]. A few traffic analysis features in LADS are similar to our method in Section V-D, although we tailored our approach to DRDoS attacks.

Kreibich et al. introduce the notion of *packet asymmetry* to detect unsolicited traffic [13]. Rather than comparing bytes, they chose to compare packet counters to capture the implicit signaling of legitimate communication. Our proposal is based on traffic volume asymmetries and can thus also detect protocol abuses in which only the bandwidth (but not the number of packets) is amplified. We have shown in Table III that – in the context of detecting DRDoS attacks – bandwidth asymmetries can be used to detect more protocols (such as CharGen, QOTD, SNMP or NetBios) than monitoring packet asymmetries.

Prior research also addressed the problem of identifying IP spoofing. For example, Eto et al. propose a statistical method to identify IP address spoofing by correlating the TTL field in the IP header with the actual hop count that a packet originating from this source should have taken [9]. Yaar et al. propose to detect DDoS traffic by tracing back the routes IP packets have taken [41]. Extending this work, Perrig et al. propose StackPi, a system to deterministically mark packets at routers to allow filtering of packets with untrustworthy routes at the recipient [25]. These approaches are complementary to our research and may help to detect the abuse of amplifiers — they cannot help to defend against DRDoS attacks from the victim's perspective, though.

Another line of research aims to detect scanning activities. Jung et al. propose Threshold Random Walk, an online detection algorithm to detect malicious port scans [11]. Sridharan proposes TAPS, which uses sequential hypothesis testing to detect hosts that exhibit abnormal access patterns in terms of destination hosts and ports [31]. Paredes-Oliva et al. compare these two portscan detection techniques and evaluate their performance in flow- and packet-based Netflow sampling settings [23]. We seek for horizontal scans in a four-week-long dataset for a *post-mortem* analysis and thus use a counting-based approach to detect scans for amplifiers.

C. DDoS Attack Analysis

Lastly, related work covers the analysis of DDoS attacks. Bailey et al. describe how darknets can help to identify scanning activities or backscatter of DoS attacks [3]. Whyte et al. propose to extend darknets by darkports, i.e., monitoring unused ports at a system to detect scanning activities [40]. Moore et al. use backscatter data to analyze tens of thousands of DoS attacks against a $/8$ network [19]. While showing the significance of the threat of DoS in general, they do not analyze DRDoS attacks in specific.

Mao et al. correlate real-world DDoS attacks found in Netflow data from a tier-1 ISP with attacks found via backscatter analysis in darknets [17]. They highlight that backscatter analysis reveals only limited insights into DDoS attacks. This supports our hypothesis that attacks where an attacker uses IP spoofing to directly attack a victim are practically irrelevant. Instead, only the *amplifiers* receive spoofed IP packets, while the *victims* receive non-spoofed responses from the amplifiers.

VII. Discussion and Future Work

By disclosing 14 vulnerable protocols we may motivate adversaries to mimic our attacks. Until now, we have not yet experienced DRDoS attacks with amplification rates as in NTP, where an attacker can use, e.g., a 1 Gb/s uplink to several-Tbit/s-large attacks. Publicly discussing these vulnerabilities may change this picture in the feature. We therefore follow the principles of responsible disclosure and alerted both, the security community (e.g., CERTs) as well as software developers of the most severely affected deployments (whenever possible). We are in the process of delivering lists of amplifiers to data clearing houses like shadowserver.org to foster alerts. Furthermore, we currently implement a live-traffic detection tool of the methods mentioned in Section IV and plan to release it to the community soon. It is unlikely that all of the vulnerable services will be hardened at some point, but we can work towards gradual improvements with these steps.

The reasons for the amplification vectors are manifold. Some of the protocols (e.g., CharGen and QOTD) have been designed in the 1980's without security in mind. Today, the legacy protocols have no real use anymore. In other cases, specific *implementations* of a protocol introduce angles for amplification. For example, public debugging functionality (like the `monlist` feature in NTP) is not officially in the protocol descriptions but added for convenience. Developers of gaming protocols or P2P protocols try to reduce the upstream traffic, as many end-users have accounts with asynchronous line speeds (i.e., higher download than upload bandwidth) [14] – resulting in possible amplification. Lastly, due to the hidden nature of P2P botnets, botmaster did not need to harden their protocols against DRDoS abuse so far and are rather concerned about protecting against takedown efforts [26].

It remains a tedious effort to design protocols that are hardened against amplification or even simple reflection. Unfortunately, we cannot claim that our list of 14 vulnerable protocols is exhaustive. In future work, we plan to automate the identification of vulnerable protocols. For example, as a starting point, one could use our methods proposed in Section IV to analyze suspicious client/server sessions with more

relaxed filtering rules. This would help to leapfrog attackers in identifying further amplification vectors.

Lastly, we left the inspection of TCP- or IP-based (instead of UDP-based) protocols for future work. Likewise, we plan to expand our experiments to IPv6, as we have found quite a few amplifiers in the IPv6 address space. IPv6 may soon become attractive to attackers, for example, because (older) filtering techniques cannot yet cope with IPv6 traffic. In addition, the IPv6 packet header is significantly larger than in IPv4, offering potential for further amplification. We plan to model these attacks in real-time DDoS attack evaluations systems like Reval [37] in the future.

VIII. Conclusion

We have conceptually and practically demonstrated that at least 14 UDP-based network protocols or service implementations are susceptible to amplification abuse. This shows that there is an urgent need to harden the protocols/implementations at least of the most severely affected services, such as DNS, NTP, SSDP or Kad. Network administrators should be prepared for DRDoS attacks that are at least a magnitude worse from what we have observed so far. We have shown, though, how amplification vulnerabilities can be effectively avoided. With our findings, we hope to start a gradual – but years-long – process of fixing the discovered protocol weaknesses. Our attack analysis and the proposed detection methods/tools support by detecting hosts under attack and identifying amplifiers that are abused.

IX. Acknowledgements

We would like to thank the anonymous reviewers, Christian J. Dietrich and John Kristoff for their constructive and valuable comments. We also thank the anonymous ISP and especially its CERT for the cooperation during this research project. We are also grateful to the CERTs that help us to alert administrators of amplifiers. Finally, we thank Jochen Schoenfelder and Christian Keil from DFN-CERT and the CarmentiS project, who provided us with one of the darknet datasets that we used in our experiments. This paper has been supported by the Federal Ministry of Education and Research (grant 16BY1201D, iAID).

References

[1] The Spoofer Project. http://spoofer.cmand.org.

[2] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos. Highly Resilient Peer-to-Peer Botnets Are Here: An Analysis of Gameover Zeus. In *Proceedings of the 8th IEEE International Conference on Malicious and Unwanted Software (MALWARE'13)*, Fajardo, Puerto Rico, USA, October 2013.

[3] M. Bailey, E. Cooke, F. Jahanian, A. Myrick, and S. Sinha. Practical Darknet Measurement. In *Proceedings of the Conference on Information Sciences and Systems(CISS)*, Princeton, NJ, March 2006.

[4] A. Büscher and T. Holz. Tracking DDoS Attacks: Insights into the Business of Disrupting the Web. In *Proceedings of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, San Jose, CA, USA, April 2012.

[5] A. Cowperthwaite and A. Somayaji. The Futility of DNSSec. In *Proceedings of the 5th Annual Symposium on Information Assurance*, Albany, NY, USA, June 2010.

[6] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescap. Analysis of a "/0" Stealth Scan from a Botnet. In *Proceedings of the Internet Measurement Conference (IMC)*, Boston, MA, USA, November 2012.

[7] K. E. Defrawy, M. Gjoka, and A. Markopoulou. BotTorrent: Misusing BitTorrent to launch DDoS attacks. In *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, Santa Clara, CA, June 2007.

[8] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *Proceedings of the 22nd USENIX Security Symposium*, Washington, D.C., USA, August 2013.

[9] M. Eto, D. Inoue, M. Suzuki, and K. Nakao. A Statistical Packet Inspection for Extraction of Spoofed IP Packets on Darknet. In *Proceedings of the Joint Workshop on Information Security*, Kaohsiung, Taiwan, August 2009.

[10] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks . In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2002.

[11] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, Oakland, CA, USA, 2004.

[12] M. S. Kang, S. B. Lee, and V. D. Gligor. The Crossfire Attack. In *Proceedings of IEEE Security and Privacy (S&P)*, San Francisco, CA, USA, 2013.

[13] C. Kreibich, A. Warfield, J. Crowcroft, S. Hand, and I. Pratt. Using Packet Symmetry to Curtail Malicious Traffic. In *Proceedings of the 4th Workshop on Hot Topics in Networks (Hotnets-VI)*, College Park, MD, USA, 2005.

[14] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating The Edge Network . In *Proceedings of the ACM Internet Measurement Conference*, Melbourne, Australia, November 2010.

[15] M. E. Donner; Prolexic. https://tinyurl.com/prolexic-167gbit, May 2013.

[16] M. Prince; CloudFlare, Inc. http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet, March 2013.

[17] Z. M. Mao, V. Sekar, O. Spatscheck, J. Van Der Merwe, and R. Vasudevan. Analyzing Large DDoS Attacks Using Multiple Data Sources. In *Proceedings of the SIGCOMM Workshop on Large-Scale Attack Defense (LSAD)*, Pisa, Italy, September 2006.

[18] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 39–53, April 2004.

[19] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, May 2006.

[20] N. Naoumov and K. Ross. Exploiting P2P Systems for DDoS Attacks. In *Proceedings of the 1st International Conference on Scalable Information Systems*, Hong Kong, May 2006.

[21] P. Ferguson, D. Senie. BCP 38 on Network Ingress

Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. http://tools.ietf.org/html/bcp38, May 2000.

[22] P.A. Porras, H. Saidi, and V. Yegneswaran. An Analysis of Conficker's Logic and Rendezvous Points. Technical report, February 2009.

[23] I. Paredes-Oliva, P. Barlet-Ros, and J. Solé-Pareta. Portscan Detection with Sampled NetFlow. In *Proceedings of the 1st Workshop on Traffic Monitoring and Analysis (TMA)*, Aachen, Germany, May 2009.

[24] V. Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. In *Computer Communication Review 31(3)*, July 2001.

[25] A. Perrig, D. Song, and A. Yaar. StackPi: A New Defense Mechanism against IP Spoofing and DDoS Attacks. Technical report, January 2003.

[26] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, and H. Bos. SoK: P2PWNED — Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2013.

[27] C. Schuba. Addressing Weaknesses in the Domain Name System Protocol. Master's thesis, Purdue University, West Lafayette, IN, USA, August 1993.

[28] V. Sekar, N. G. Duffield, O. Spatscheck, J. E. van der Merwe, and H. Zhang. LADS: Large-scale Automated DDoS Detection System. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, Boston, MA, USA, May 2006.

[29] K. C. Sia. DDoS Vulnerability Analysis of BitTorrent Protocol. Technical report, Spring 2006.

[30] S. M. Specht and R. B. Lee. Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. In *Proceedings of the International Conference on Parallel and Distributed Computing (and Communications) Systems (ISCA PDCS)*, San Francisco, CA, September 2004.

[31] A. Sridharan. Connectionless Port Scan Detection on the Backbone. In *Proceedings of the Malware Workshop (held in conjunction with IPCCC)*, Pheonix, AZ, USA, April 2006.

[32] A. Studer and A. Perrig. The Coremelt Attack. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, Saint Malo, France, September 2009.

[33] X. Sun, R. Torres, and S. Rao. DDoS Attacks by Subverting Membership Management in P2P systems. In *Proceedings of the 3rd IEEE Workshop on Secure Network Protocols (NPSec)*, Göttingen, Germany, October 2007.

[34] X. Sun, R. Torres, and S. Rao. On the Feasibility of Exploiting P2P Systems to Launch DDoS Attacks. In *Journal of Peer-to-Peer Networking and Applications*, volume 3, pages 36–51, March 2010.

[35] X. Sun, R. Torres, and S. Rao. Preventing DDoS Attacks on Internet Servers Exploiting P2P Systems. In *Elsevier Journal on Computer Networks (COMNET)*, volume 54, October 2010.

[36] T. Brewster (TechWeek). http://www.techweekeurope.co.uk/news/zimbabwe-election-cyber-attacks-123938, August 2013.

[37] R. Vasudevan, Z. M. Mao, O. Spatscheck, and J. van der Merwe. Reval: A Tool for Real-time Evaluation of DDoS Mitigation Strategies. In *Proceedings of USENIX Annual Technical Conference (ATC)*, Boston, MA, USA, May 2006.

[38] P. Vixie and V. Schryver. DNS Response Rate Limiting (DNS RRL). http://ss.vix.su/~vixie/isc-tn-2012-1.txt.

[39] X. Wang and M. K. Reiter. Mitigating Bandwidth-Exhaustion Attacks Using Congestion Puzzles. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, New York, NY, USA, October 2004.

[40] D. Whyte, P. C. v. Oorschot, and E. Kranakis. Tracking Darkports for Network Defense. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, FL, USA, December 2007.

[41] A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, Oakland, CA, USA, May 2003.