# Amplification from Hell

Roee Shlomo

November 17, 2014

**Abstract**

This document is a summary and discussion around several papers ([4] [1] and [2]) on the topic of DRDoS amplification attacks. In recent years such attacks became common and easily exceed hundreds of Gbps of attack volume, leading to serious downtime of victim systems. In this paper we discuss how such attacks are performed, list and discuss the amplification characteristics of 14 UDP-based protocols that were identified as vulnerable for abuse and show how TCP can be abused for amplification. Finally, we discuss ways to lower the threat, counter and avoid amplification attacks in different ways.

# 1 INTRODUCTION

## 1.1 Background

A Denial of Service attack is a malicious attempt to make a server or a network resource unavailable to users. A Distributed DoS attack involves multiple systems (e.g. using a bot network). Such attacks can target either the Application layer by exhausting application resources or the Network layer in various forms including bandwidth exhaustion by flooding victim servers, link flooding (Coremelt [5], Crossfire [3]) and control plain attacks (CXPST).

Modern DDoS attacks usually flood victim servers by abusing UDP based network protocols to employ either reflective or, even better, amplification attacks. We call such attacks DRDoS (Distributed Reflective Denial of Service) attacks. In this type of attacks the attacker abuses public servers that use a vulnerable protocol by sending requests with a spoofed source IP address, i.e. that of the victim. The victim those receives incoming data from the servers and not directly from the attacker (reflection). Attackers often use request types that result with much larger responses, leading to **amplification** of the attack volume. In this paper we focus on Amplification attacks using either TCP abuse or vulnerable UDP-based protocols abuse.

## 1.2 Threat Model

The threat model are distributed and reflective denial of service attacks in which an attacker **A** attempt to exhaust the bandwidth of victim **V** by abusing a set of amplifiers **Mp** that response to valid requests of protocol **P**. Attacker A sends attack traffic to several amplifiers (thus distributed) using a spoofed IP address of victim V. The amplifiers in turns send response traffic to victim V leading to incoming attack traffic from several sources. Figure 1 illustrates the threat model where an attacker abuses several amplifiers to exhaust the bandwidth of a victim. Most DRDoS attacks today use similar setup.

Victim V is a single host identified by a single IP address. There are no assumptions as for A's bandwidth limitations and A can be anything from a single host to the controller of a large botnet system, we show what can be achieved for any uplink bandwidth limit. There is an assumption that A can send spoofed packets without them being filtered (e.g. by its ISP), we will discuss this assumption in detail later.

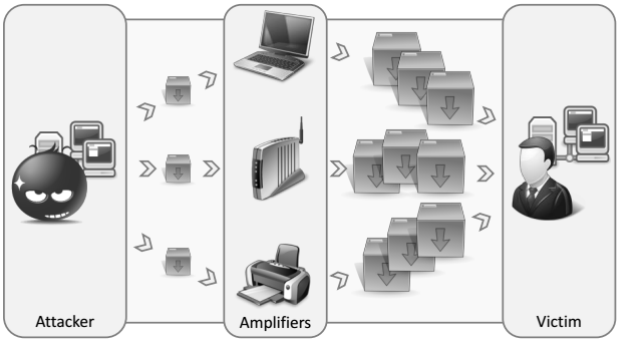


Figure 1: Threat model illustration - An attacker sends requests to amplifiers with the victim's IP address as IP packet source. In turn, the amplifiers send (potentially multiple) large responses to the victim.

## 1.3 Definitions

Multiple measurement of amplification can be used and we define the ones we will use in this document here.

**Bandwidth amplification factor (BAF)** the bandwidth multiplier in terms of number of payload bytes that an amplifier sends to answer a request, compared to the number of payload bytes of the request itself. Notice that this measurement excludes headers so it would hold for any lower level protocol changes (e.g. change from IPv4 and IPv6). I argue that such a measurement does not truly show the amplification factor as in some protocols the header sizes are not negligible compared to the payload making the BAF much higher than the actual amplification factor. It is however a measurement we can look at to identify good amplification attack vectors and an amplification factor that takes headers into account can be calculated from it given protocol specific information. $BAF = \frac{len(Payload)_{Mp\ to\ V}}{len(Payload)_{A\ to\ Mp}}$

**Packet amplification factor (PAF)** the multiplier in terms of number of IP packets that an amplifier sends to answer a request, compared to the number of IP packets of the request itself. This measurement suits some protocols better and specifically is ideal for the case of TCP amplification. $PAF = \frac{\#packets_{Mp\ to\ V}}{\#packets_{A\ to\ Mp}}$

**Headers inclusive BAF (HiBAF)** HiBAF is similar to BAF but takes into account Ethernet, IP and transport protocol headers. $HiBAF = \frac{len(Packet)_{Mp\ to\ V}}{len(Packet)_{A\ to\ Mp}}$

This is probably the best definition. Unfortunately, the original papers often omit it, especially for UDP-based protocols. I will add this information were possible to give a clearer picture of the amplification factor. HiBAF is estimated from the BAF using the following formula (where $42 = 14_{ETH\_HDR} + 20_{IP\_HDR} + 8_{UDP\_HDR}$ and 64 is the minimal Ethernet frame size in bytes):

$$\frac{len(ReqestPayload) \times BAF + 42 \times PAF}{max(len(ReqestPayload) + 42, 64)}$$

## 1.4 Acquiring a set of amplifiers

As a preparation step, $A$ needs to acquire a set of amplifiers $Mp$ to be used in the attack. Assuming $A$ knows protocol $P$ it is fairly easy for attacker to discover a set of amplifiers so in

this paper that's a non-issue. Different protocols require different methods though:

1. Amplifiers of fixed port network services (e.g. DNS open resolvers) can be discovered by a scanning the IPv4 address space.
2. Amplifiers of P2P based services (e.g. Kad) can be discovered using crawling.
3. Some protocols are designed in a way such that each machine is registered at a master server that allows querying for its members. That's the case for common game servers (e.g. Steam).

# 2 UDP AMPLIFICATION

In his work[4], Rossow identified 14 UDP protocols that allow amplification attacks. Those include several network services (SNMPv2, NTP, DNS, NetBios, SSDP), legacy protocols (CharGen, Quote Of The Day), P2P networks (BitTorrent DHT, eMule Kad), game servers (Quake 3, Steam) and even malicious botnets that can be abused by anyone due to their P2P architecture (ZeroAccess, Sality, GameOver).

The design characteristic that allow amplification abuse in these protocols is that there is no source IP verification and a single request results in a response without requiring any handshake process. Note that to be considered as a vulnerable for amplification protocol, it suffices that just part of the protocol allows request-response scheme in which no handshake is required and the response volume is larger than the request volume.

Notice that in our attack model the attack impact is directly related to the number of available amplifiers and to their AF. The attack bandwidth is constrained by the total bandwidth volume produced by the set of available amplifiers. Table 1 lists the UDP protocols identified by Rossow with their PAF values, BAF values for subsets of all, top 50% and top 10% amplifiers, an estimated number of available amplifiers and the method used for abuse. In this paper we only review some of these protocols. We first only consider protocols that allow BAF above 50x. Looking at the BAF for the top 10% amplifiers of each protocol we can observe that many protocols allow BAF of less than 50x. The exceptions are DNS, NTP, SSDP, Quake 3 and the legacy protocols CharGen and QOTD (for the entire set of amplifiers). We further filter on protocols that have a large set of amplifiers. This excludes Quake 3 and the legacy protocols from our list and we end up with DNS, NTP and SSDP. Note though that CharGen and QOTD can and actually are being used in large scale DRDoS attacks, either alone or as an addition to another protocol abuse, so closing these services should be a priority to the networking security community.

DNS and NTP were recently abused to perform some of the largest DDoS attacks. SSDP DDoS attacks are on the rise in current days (PLXsert threat advisory, Akamai, October 15, 2014). In this section we review the vulnerability of these three protocols in detail.

## 2.1 DNS (Domain Name System)

The DNS infrastructure contains several types of building blocks including stub resolvers, DNS resolvers and authoritative name servers. We distinguish between two types of DNS amplifiers, one abusing DNS resolvers and the other abusing authoritative NS.

First, the well-known problem of Open DNS Resolvers is a major DDoS threat. Open DNS resolvers are DNS resolvers that allow recursive queries by any client instead of only responding

| Protocol | #Amplifiers | BAF | | | PAF | Method |
|---|---|---|---|---|---|---|
| | | All | Top 50% | Top 10% | | |
| SNMP v2 | 4,832,000 | 6.3 | 8.6 | 11.3 | 1.00 | GetBulk request |
| NTP | 1,451,000 | 556.9 | 1083.2 | 4670.0 | 3.84 | Request client statistics |
| DNS NS | 255,819 | 54.6 | 76.7 | 98.3 | 2.08 | ANY lookup at author. NS |
| DNS OR | 7,782,000 | 28.7 | 41.2 | 64.1 | 1.32 | ANY lookup at open resolv. |
| NetBios | 2,108,000 | 3.8 | 4.5 | 4.9 | 1.00 | Name resolution |
| SSDP | 3,704,000 | 30.8 | 40.4 | 75.9 | 9.92 | SEARCH request |
| CharGen | 89,000 | 358.8 | n/a | n/a | 1.00 | Character generation request |
| QOTD | 32,000 | 140.3 | n/a | n/a | 1.00 | Quote request |
| BitTorrent | 5,066,635 | 3.8 | 5.3 | 10.3 | 1.58 | File search |
| Kad | 232,012 | 16.3 | 21.5 | 22.7 | 1.00 | Peer list exchange |
| Quake 3 | 1,059 | 63.9 | 74.9 | 82.8 | 1.01 | Server info exchange |
| Steam | 167,886 | 5.5 | 6.9 | 14.7 | 1.12 | Server info exchange |
| ZAv2 | 27,939 | 36.0 | 36.6 | 41.1 | 1.02 | Peer list and cmd exchange |
| Sality | 12,714 | 37.3 | 37.9 | 38.4 | 1.00 | URL list exchange |
| Gameover | 2,023 | 45.4 | 45.9 | 46.2 | 5.39 | Peer and proxy exchange |

Table 1: Vulnerable UDP protocols and their amplification characteristics. The number of amplifiers is an estimate. Particularly, when IPv4 address space scanning is being used, only a subset of 1M advertised addresses is scanned and the result is multiplied accordingly for an estimate. When crawling is used we only run the crawler for 1 hour. The value n/a indicates the number of amplifiers in our actual data set is too small for the calculation to be meaningful.

| Protocol | All | | | Top 50% | | | Top 10% | | | PAF | PL(B) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAF | HiBAF | Ratio | BAF | HiBAF | Ratio | BAF | HiBAF | Ratio | | |
| SNMP v2 | 6.3 | 4.5 | 0.71 | 8.6 | 6.03 | 0.7 | 11.3 | 7.81 | 0.69 | 1 | 82 |
| NTP | 556.9 | 72.13 | 0.13 | 1083.2 | 137.92 | 0.13 | 4670 | 649.38 | 0.14 | 3.84 | 8 |
| DNS NS | 54.6 | 20.13 | 0.37 | 76.7 | 27.73 | 0.36 | 98.3 | 35.16 | 0.36 | 2.08 | 22 |
| DNS OR | 28.7 | 10.73 | 0.37 | 41.2 | 15.03 | 0.36 | 64.1 | 22.9 | 0.36 | 1.32 | 22 |
| SSDP | 30.8 | 23.61 | 0.77 | 40.4 | 29.91 | 0.74 | 75.9 | 53.19 | 0.7 | 9.92 | 80 |
| CharGen | 358.8 | 6.26 | 0.02 | n/a | n/a | n/a | n/a | n/a | n/a | 1 | 1 |
| QOTD | 140.3 | 2.85 | 0.02 | n/a | n/a | n/a | n/a | n/a | n/a | 1 | 1 |
| BitTorrent | 3.8 | 3.16 | 0.83 | 5.3 | 4.23 | 0.8 | 10.3 | 7.79 | 0.76 | 1.58 | 104 |
| Kad | 16.3 | 7.95 | 0.49 | 21.5 | 10.32 | 0.48 | 22.7 | 10.86 | 0.48 | 1 | 35 |
| Quake 3 | 63.9 | 15.64 | 0.24 | 74.9 | 18.22 | 0.24 | 82.8 | 20.07 | 0.24 | 1.01 | 15 |
| Steam | 5.5 | 2.75 | 0.5 | 6.9 | 3.28 | 0.48 | 14.7 | 6.19 | 0.42 | 1.12 | 25 |
| ZAv2 | 36 | 9.67 | 0.27 | 36.6 | 9.82 | 0.27 | 41.1 | 10.94 | 0.27 | 1.02 | 16 |
| Gameover | 45.4 | 27.52 | 0.61 | 45.9 | 27.8 | 0.61 | 46.2 | 27.97 | 0.61 | 5.39 | 52 |

Table 2: Comparison of BAF values to HiBAF values. For top 10% of NTP amplifiers the HiBAF estimation formula assumes PAF of 100 as observed in experiments; for all others the specified average PAF value is used. NetBios and Sality omitted due to lack of information. Last column specifies the UDP payload size of a request; SMMPv2 and SSDP payloads were observed using Wireshark; BitTorrent payload size is described in protocol specification; DNS payload size is determined assuming 8B length domain name is used; for all other protocols the request payload size is specified in the original papers.

to authorized clients. By scanning the IPv4 address space one can acquire a set of millions of open resolvers, we named this set $DNS_{OR}$ and you can notice this set is by far the largest. To abuse these amplifiers an attacker would typically send requests of type ANY to ask the DNS resolver for all information it currently has about the domain in question. This allows the attacker to maximize the response size and increase the AF. Traditionally, DNS messages are limited to 512B and switch to TCP connection for delivery of larger messages. Considering we must avoid the IP address verification at the TCP handshake step, this poses a limitation on the response size and thus on the BAF. Luckily for the attacker, the DNS extensions, named EDNS0, allow exceeding this limit and sending responses of size up to 4096B over UDP. The attacker can even control the response size and force it to its maximum by configuration a name server of a domain she controls to return exactly 4KB as a response, taking advantage of the cache in open resolvers so that the amplifiers will cache the big response according to the TTL value and therefore the attacker's NS would only face little load. Keep in mind that such a setup exposes the attacker's domain to the victim but it was actually observed in real attacks. As shown in Table 1 the BAF for $DNS_{OR}$ varies from 28.7 to 64.1, depending on the queried domain length and maximal response size supported by the open resolver. Many open resolvers did not support EDNS0 and thus the response got chopped to 512B.

The second type of DNS amplifiers are authoritative NS that support DNSSEC, we call this set $DNS_{NS}$. In DNSSEC the nameserver responds with an additional resource record named RRSIG that contains a 1024bit long signature used for authentication, integrity and proof of non-existence. In the tests, 1404 name servers that support DNSSEC have been used. Notice that although proposed in 1997, DNSSEC is not widely deployed at the time of this writing, yet the number of authoritative name servers with DNSSEC support is increasing gradually. As shown in Table 1 the BAF observed for $DNS_{NS}$ is 54.6 and can reach 98.3 for the top 10% of amplifiers. It was observed that most amplifiers in this set actually support EDNS0.

The HiBAF values for $DNS_{OR}$ and $DNS_{NS}$ are not specified in the original papers. Table 2 shows the estimated HiBAF for 12 of the UDP-based protocols. For $DNS_{OR}$ the HiBAF varies from 10.73 to 22.9. For $DNS_{NS}$ the HiBAF varies from 20.13 to 35.16. A ratio of 1/3 between HiBAF and BAF. A minimal DNS request contains 54B of headers (layers 2-7), 2B for type ANY and a variable length domain name. Assuming the attacker can use requests of size 64B (minimum Ethernet frame size), the maximum possible HiBAF is 8.65 in case EDNS0 isn't supported and 64.65 otherwise.

Abusing Open DNS resolvers is a well-known method used in actual amplification attacks. In March 2013, a 300Gbps attack was performed targeting Spamhaus (though not directly) in what press called "The DDoS That Almost Broke the Internet" as the attack reached tier-1 providers. This attack was launched by an attacker controlling 10 compromised servers on 3 networks that allow IP spoofing, generating 9Gbps DNS requests to 32,000 open DNS resolvers

We will later discuss means to migrate and avoid amplification DDoS attacks in general but for the specific case of $DNS_{OR}$, lowering the number of open resolvers is trivially the first step and indeed network operators have become aware of this kind of abuse and the number of open resolvers is gradually decreasing. The Open Resolver Project published a list of millions of open resolvers in an effort to shut them down. Latest report from 11/16/2014 indicates almost 20 million resolvers can be abused. Network administrators should make sure their recursive resolvers are configured to only give service to restricted IP ranges, of their enterprise or customers. $DNS_{NS}$ though don't have a simple configuration correction, it is commonly suggested to enforce rate limiting and we will later discuss how effective rate limiting actually is.

## 2.2 NTP (Network Time Protocol)

NTP stands out as the most vulnerable protocol among the 14 UDP protocols identified. Not only that its BAF varies from 556 to no less than 4670, NTP servers also have very low IP address churn. Therefore, NTP is very attractive for attackers.

In December 2013 a 100Gbps attack against game services like EA, Riot Games, Blizzard, Valve, and many others was reported to abuse NTP monlist. In February 2014, CloudFlare observed a 400Gbps attack against a French hosting provider. This attack is the largest DDoS attack observed so far and have the attacker had more resources the impact of the attack could have been much higher. It is therefore obvious that NTP vulnerabilities pose a serious threat.

An attacker can take advantage of the NTP **monlist** feature. Although this feature is not part of the NTP protocol itself, it's supported by many NTP servers as a debugging feature. The monlist command requests a list of recent clients that contacted the server. This list can hold up to 600 clients leading to a maximal payload of about 44kB. It is possible to disable this feature for unauthorized users and in recent ntpd versions this feature is protected with session handshake. Network administrators that are not aware of the threat use earlier versions of ntpd with default configuration that does not protect against monlist abuse. A second attack vector using NTP is abusing its **version** feature which reveals system fingerprint and allows BAF of about 24.

The maximum HiBAF of NTP monlist **should** normally be a around 206 (As observed using Wireshark, request size is 234B and response spans 100 packets of 482B each). The authors mention a short form of the monlist command exists that require just 8B of data (NTP header). Considering the minimum size of 64B for Ethernet frames, this leads to a maximal HiBAF of 753 in theory. As shown in Table 2 the HiBAF of NTP observed in the scanning test varies from 72.1 to 649.4.

NTP has been used as a case study by the authors of paper "Exit from Hell" [1]. In a large scale campaign they have launched a global notification procedure to alert NTP administrators about the amplification problems towards the goal of reducing the number of NTP amplifiers. As a result of collaborating with security organization and alerting Cisco, which was found to be the vendor behind many vulnerable devices, public advisories of CERT-CC, MITRE and Cisco's PSIRT were published. These advisories describe how to disable the monlist and version features. In addition, the authors distributed, among trusted institutions, a list of IP addresses of devices vulnerable to monlist abuse. They also notified the NTP Pool Project about misconfigured ntp servers in their public pool and also synchronized their notifications with the Open NTP Project, a project with a similar task, to start announcements simultaneously. As can be seen, their focus was the monlist abuse as it is by far more severe.

To observe the effectiveness of this campaign weekly scans were performed. Figure 2 shows the number of amplifiers per week and mark important events. As can be seen, the peak was on December 15, 2013, and 1,651,199 $NTP_{MON}$ amplifiers were observed. The number of amplifiers started to drop (15.4%) right after the release of the MITRE advisory and the release of the first, partial, list of IP addresses. A second drop (43.4%) happened after the release of two complete lists of IP addresses and the publication of CERT-CC and PSIRT advisories. A steady decrease continued along with weekly notifications published by the authors and on February 24, 2014, only 126,080 $NTP_{MON}$ amplifiers were found. An additional scan was performed separately on Jun 20, 2014, and only 87,463 vulnerable hosts were found. This concludes an impressive decrease of about 92.2%.

So what happened here? Looking at the AS distribution indicates that most vulnerable hosts were closed using network level packet filtering. About half of the 96 ASes that had at

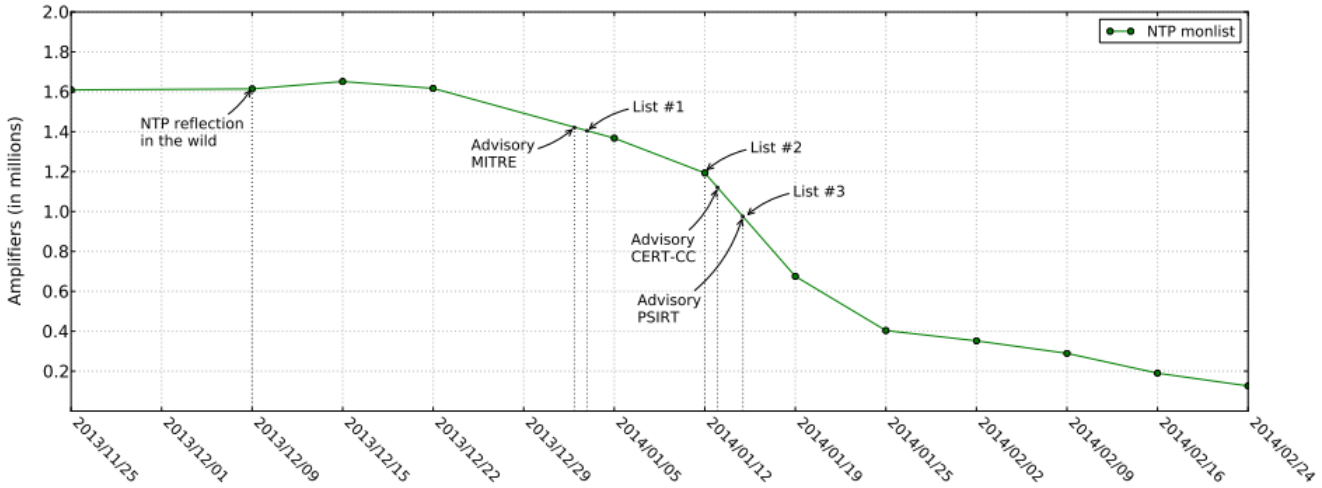


Figure 2: Trend of NTP monlist amplifiers.

least 1,000 amplifiers closed over 95% of the amplifiers. Also, 73 ASes that were observed to have over 100 NTP servers in some week had none open in following week, indicating packet filtering was indeed employed.

Note that in the 400Gbps attack mentioned before, the attacker used 4,529 NTP servers running on 1,298 different networks. On average, each of these servers sent 87Mbps of traffic to the intended victim on CloudFlare's network. Remarkably, it is possible that the attacker used only a single server running on a network that allowed source IP address spoofing to initiate the requests. Those numbers suggest that the number of vulnerable hosts must still be lowered.

## 2.3 SSDP

SSDP comes enabled on millions of home and office devices, including routers, media servers, web cams, smart TVs and printers. The protocols allow devices to discover each other on a network, establish communication and coordinate activities.

To abuse an SSDP device an attacker issues a discovery SEARCH request. All UPnP enabled hosts that receive the request then respond with one reply packet per service they have configured. The BAF varies due to the difference in service names lengths and the difference in number of services offered by the host device. On average, a BAF of 30.8 and a PAF of 9.91 was achieved. Using the top 10% amplifiers an attacker can obtain a BAF of 75.9. The HiBAF of SSDP varies from 23.62 to 53.16 (A HiBAF to BAF ratio of 0.77 and 0.7 respectively).

As mentioned above, Akamai PLXsert recently released a threat advisory addressing this issue. According to Akamai representative, the number of UPnP devices that will behave as open reflectors is vast, and many of them are home-based Internet-enabled devices that are difficult to patch. Packet filtering seems like the best solution for this protocol abuse and we will later see if it's possible. PLXsert began seeing attacks from UPnP devices in July 2014 and they found 4.1 million Internet-facing UPnP devices are potentially vulnerable to amplification abuse, a similar number to the estimate in this paper. PLXsert will share the list of potentially exploitable devices to members of the security community in an effort to collaborate with cleanup and mitigation efforts of this threat.

# 3 TCP AMPLIFICATION

TCP uses session handling and thus should prevent amplification. During the handshake a client sends SYN and expects a SYN-ACK response. Only on recipient of the SYN-ACK the client can start sending data along with an ACK, which is not the case for our attacker that spoof the IP address and never receive the SYN-ACK. Notice that the SYN and SYN-ACK packets are of the same size and thus TCP does not allow amplification, or at least should not allow it on proper operation. However, different behaviors in regard to responses for TCP SYN packets were observed in real life scans and indicated amplification can actually be achieved using TCP SYN packets. In this section we discuss the scanning performed and results found.

## 3.1 Scanning setup

An initial scan was performed over 20M random IP addresses for 13 common TCP based protocols. After the initial scan a full scan over the entire IPv4 address space was performed for the protocols that were estimated to have over 5,000 amplifiers in the entire IPv4 address space (i.e. FTP, HTTP, NetBIOS, SIP, SSH and Telnet). For every protocol the scanner sends a single SYN packet to the target host and records the responses. The scanner does **not** complete the handshake nor does it send RST response. Note that only hosts that amplify by a factor above 20 are considered. Also note that in this test the amplification factor considers not only the payload but also TCP, IP and Ethernet headers.

## 3.2 Results

| | 20 million random hosts | | | Estimation (IPv4) |
|---|---|---|---|---|
| Protocol | # Responsive | # Amplifiers | Amplifier Ratio | # Amplifiers |
| *FTP* | 705,371 | 13,701 | 1 : 51 | 2,945,715 |
| *HTTP* | 715,354 | 1,746 | 1 : 409 | 375,390 |
| *IMAP* | 683,567 | 9 | 1 : 75,951 | 1,935 |
| *IPP* | 702,590 | 19 | 1 : 36,978 | 4,085 |
| *IRC* | 648,688 | 7 | 1 : 92,669 | 1,505 |
| *MySQL* | 672,336 | 8 | 1 : 84,042 | 1,720 |
| *NetBIOS* | 517,482 | 44 | 1 : 11,760 | 9,460 |
| *NNTP* | 667,598 | 8 | 1 : 83,449 | 1,720 |
| *POP3* | 689,716 | 7 | 1 : 98,530 | 1,505 |
| *SIP* | 711,210 | 87 | 1 : 8,174 | 18,705 |
| *SMTP* | 674,815 | 12 | 1 : 56,234 | 2,580 |
| *SSH* | 657,916 | 384 | 1 : 1,713 | 82,560 |
| *Telnet* | 575,067 | 9,315 | 1 : 61 | 2,002,725 |

Table 3: Number of potential amplifiers with an amplification factor >20 based on scans of 20 million hosts

| | | # Amplifiers with amplification factor | | | | | |
|---|---|---|---|---|---|---|---|
| Protocol | # Responsive | > 20 | > 50 | > 100 | > 500 | > 1,000 | > 2,500 |
| *FTP* | 152,026,322 | 2,913,353 | 3,500 | 1,868 | 1,032 | 937 | 847 |
| *HTTP* | 149,521,309 | 427,370 | 15,426 | 6,687 | 1,596 | 649 | 347 |
| *NetBIOS* | 82,706,193 | 12,244 | 2,449 | 1,463 | 873 | 811 | 783 |
| *SIP* | 154,030,015 | 22,830 | 5,158 | 3,913 | 3,289 | 3,123 | 2,889 |
| *SSH* | 141,858,473 | 87,715 | 4,611 | 2,141 | 1,275 | 1,176 | 1,082 |
| *Telnet* | 126,133,112 | 2,120,175 | 16,469 | 7,147 | 2,008 | 1,393 | 994 |

Table 4: Number of potential amplifiers per protocol based on scans in the entire IPv4 address space

Tables 3 and 4 show the results of the initial and full scans, respectively. In the initial scan FTP and Telnet had the highest number of amplifiers while most others had just a few amplifiers at all. The full scan considered only 6 of these protocols as indicated above. Almost 3 million FTP hosts and over 2 million Telnet hosts allow amplification factor of above 20. Notice that SIP has about 2,800 hosts that allow amplification of above **2,500**. There is an overlap between the protocols in regards to the actual vulnerable hosts running them. Table 5 shows this overlap and as you can notice the largest overlay is between NetBIOS and SIP (above 50%). In total, 4.8 million distinct IP addresses of amplifiers were found for the six protocols.

| Protocol | Intersection (in %) | | | | | |
|---|---|---|---|---|---|---|
| | *FTP* | *HTTP* | *NetBIOS* | *SIP* | *SSH* | *Telnet* |
| *FTP* | - | 4.5 | 0.1 | 0.2 | 0.6 | 19.2 |
| *HTTP* | 30.8 | - | 0.6 | 1.1 | 4.3 | 26.7 |
| *NetBIOS* | 20.9 | 21.9 | - | 53.8 | 14.7 | 22.5 |
| *SIP* | 21.9 | 21.4 | 28.9 | - | 22.5 | 26.0 |
| *SSH* | 19.8 | 21.1 | 2.1 | 5.9 | - | 21.0 |
| *Telnet* | 26.3 | 5.4 | 0.1 | 0.3 | 0.9 | - |

Table 5: Intersection of potential amplifiers

| Protocol | SYN/ACK | | PSH | | RST | |
|---|---|---|---|---|---|---|
| | # Ampl. | AF | # Ampl. | AF | # Ampl. | AF |
| *FTP* | 2,907,279 | 22x | 274 | 103x | 5,577 | 53,927x |
| *HTTP* | 421,487 | 60x | 241 | 147x | 3,411 | 432x |
| *NetBIOS* | 8,863 | 54x | 64 | 71x | 3,087 | 78,042x |
| *SIP* | 16,496 | 1,596x | 2 | 696x | 6,306 | 32,411x |
| *SSH* | 81,256 | 80x | 391 | 57x | 5,889 | 29,705x |
| *Telnet* | 2,112,706 | 28x | 2,353 | 3,272x | 4,242 | 79,625x |

Table 6: Number of vulnerable hosts and average HiBAF (shown as AF) per protocol and amplification type

## 3.3 Amplification types

Three main categories of unexpected SYN response behaviors were observed during the scans. Table 6 shows the results in regards to these amplifier types, per protocol.

1. **SYN/ACK:** Amplifiers that aggressively retransmit SYN-ACK packets. The vast majority of amplifiers found are of this type. On average it allows amplification factor of 80 and for SIP it can reach a factor of 1,596.

2. **PSH:** Amplifiers that send data using PSH packets even though the three way handshake never completed. There are relatively a small number of amplifiers of this type, though the average amplification factor is in fact higher than that of the SYN/ACK type.

3. **RST:** Amplifiers that send a flood of RST segments to refuse the connection attempt. A few thousands of amplifiers found but the more interesting point is the large amplification factor. As an example an attacker can use the 4,242 Telnet amplifiers found to achieve an amplification factor of 79,625. The RST amplifiers of most protocols have much higher traffic volume than that of SYN/ACK amplifiers even though their set size is much smaller. In example using the set of FTP SYN/ACK amplifiers resulted in 3.2GB of traffic volume while the FTP RST amplifier resulted in 15.1GB of traffic volume at the same amount of time.

## 3.4 Limitations

It was found that if a victim sends a RST as a response to unexpected SYN-ACK segment, the amplification of SYN/ACK amplifiers can be eliminated (99.9% decrease in the number of available amplifiers). The same approach decreases RST and PSH amplifiers though not by a significant factor. A similar solution suggests sending ICMP port unreachable, in a test on a 3500 amplifiers set this approach resulted in a decrease of 78.8% in the number of amplifiers. An attacker can avoid that by spoofing a none-existing IP address on the victim's network, so that no host would respond with a RST or ICMP message. Additionally, it can be assumed that during a large scale attack the victim won't be able to send outgoing packets at all making this a non-issue.

A more serious limitation of TCP handshake amplification is the packet frequency. The impact of an attack is affected by the number of packets that reach the target host simultaneously. To measure it, the number of packets transmitted by each amplifier within 10, 30 and 60 seconds was recorded. The results indicate that SYN/ACK amplifiers are not suitable for amplification attacks as only a few packets arrived in 60 seconds interval, and actually the most

was observed for NetBIOS and was just 22 packets. The PSH amplifiers showed similar results with the exception of the Telnet protocol in which 52, 154 and 277 packets were observed in the specified intervals. The RST amplifiers show a completely different picture and for NetBios almost 1,000 packets arrived within 10 seconds and over 5,000 packets arrived within 60 seconds. For Telnet about 4,250 packets arrived within 60 seconds, on average.

To check the feasibility of attacks using TCP handshake amplification another test was performed using hosts that run Telnet services. A random subset of 100,000 SYN/ACK amplifiers and all PSH and RST amplifiers, previously discovered in the full IPv4 address space scan, were chosen. In total only 62,736 SYN/ACK, 2203 PSH and 1593 RST amplifiers were still responsive. For SYN/ACK, transmitting 1 SYN to each of the 62,736 amplifiers resulted in just 34MB of traffic volume, sending 10 SYN segments increased the volume to just 76MB, an increase of factor x2.2. For PSH, transmitting 1 SYN to each of the 2203 PSH amplifiers resulted in 11.2MB of traffic volume, sending 10 SYN segments increased the volume to 110.8MB, an increase of factor x10. For RST, transmitting 1 SYN to each of the 1593 RST amplifiers resulted in 89.6MB of traffic volume, sending 5 SYN segments increased the volume to 392.4MB, an increase of factor x4.4, sending 10 SYN segments resulted in further increase by factor of x2 to 789.2MB of network traffic recorded within 60 seconds.

The results indicate that SYN/ACK amplifiers are not suitable for large scale amplification attacks. The authors argue that PSH and RST amplifiers can be abused to launch a large scale DDoS attack.

# 4 COUNTERMEASURES

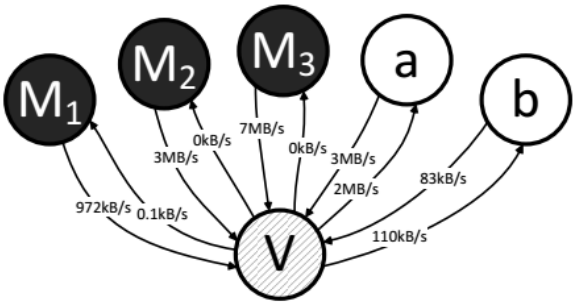


Figure 3: A DRDoS attack against a host V with three attacking amplifiers (M1, M2, M3) and two legitimate hosts (a, b). The arrows show the average bandwidth per communication stream.

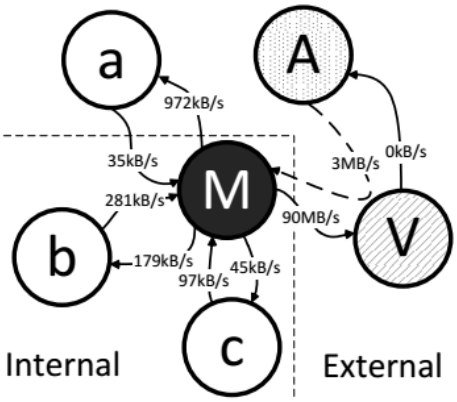


Figure 4: An scenario where A abuses M as amplifier to attack V , while M also has two legitimate clients in the internal network (b, c) and one external client (a)

## 4.1 Attack detection

We first want to check if an attack can easily be detected by an ISP, and specifically whether it's possible to detect attack victims and detect attack amplifiers in a simple way. Although the approach shown here is offline it has been used to detect and analyze real world attacks and can further help with fixing and closing amplifiers. It remains in question whether such approach can be used online reliably. We do not consider anything other than detection here.

DRDoS victims will receive large amounts of traffic from amplifiers, while they have never requested data from the amplifiers. See Figure 3 for illustration of a typical amplification attack scenario. Using this observation and given access to view the entire network (from the ISP's perspective) we could build some rules to detect attack victims. We consider only protocols with fixed port for simplification. We define $pairflow := < C_{IP}, S_{IP}, S_{port}, B_{2s}, B_{2c}, t >$ as the connection flow between a client and a server where $C_{IP}$ is a client IP address, $S_{IP}$ and $S_{port}$ identify the server by its IP and port, $B_{2s}$ is the number of bytes sent from client to server, $B_{2c}$ is the number of bytes sent to client and $t$ is the duration of the flow which we can use to calculate the average bandwidth of the flow. We assume we have access to most pairflows in the network and apply a heuristic to filter the dataset and find clients that are potential victims. Note that we discard all pairflows with servers that are within the network of the ISP, as we do not consider threats from within the network. A first filter is applied to focus on pairflows exceeding a traffic threshold $T_B$ and a second filter to focus on client that received significantly more traffic than they sent to the server so the ratio $r_f = \frac{f.B_{2s}}{f.B_{2c}}$ is higher than a threshold $T_r$ (note that usually $r_f$ should be $\infty$). We set $T_B$=100,000 bytes and $T_f$=1000 to identify victims and use it to identify attacks in a Netflow dataset obtained from a large European ISP comprising about 1 million end users (consumer and business) and spans for 12 days. Fifteen attacks were identified, the largest spanned 711 MBit/s from 330 amplifiers and lasted for three hours. Of the attacks 11 abused DNS and 4 abused CharGen, we notice that some of these overlap in time and target the same victims, meaning both DNS and CharGen were abused together. The results were verified by discussing them with the ISP's CERT and indeed all the DNS based attack victims were known to the CERT due to basic alerting systems, 3 of the CharGen attacks followed earlier attacks on the same victims so the victims were known to the CERT and the fourth CharGen attack severely interfered with a DSL-connected customer. Also note that the CERT was not aware of any other attacks, which indicates a small chance of false negatives.

To identify legitimate services that are abused as amplifiers during a DRDoS attack we can use a similar approach. However the filtering thresholds must be more relaxed and we expect some false positives because the case of amplifiers detection is more difficult than victim detection, as amplifiers are usually servers that legitimately both send and receive large amount of data. Figure 4 illustrates a typical amplification attack scenario from the amplifier perspective. Notice that it seems as if we could use the same approach we used for victim detection if only we use different thresholds. This approach may lead to false positives though since legitimate traffic can sometimes be similar to amplification abuse, e.g. when requesting DNSSEC records from a DNS resolver. We set $T_B$=10,000,000 (10MB) and $T_r$=5.0 to focus on aggressive attacks. We further discard all pairflows with bandwidth below 10Kbps to filter pairflows with low volume communication. This approach was again used on the Netflow dataset and flagged 143 pairflows as suspicious. The results identified 6 open DNS resolvers abused in 55 attacks, 1 authoritative name server abused in 3 attacks, 4 CharGen servers abused to attack 57 victims and even found alerts for a potential abuse of three Steam game servers. Notice however that four closed DNS recursive resolvers were flagged as suspicious, in this case due to them trying to resolve domains from authoritative name servers that suffered extensive packet loss, hence false positives are indeed possible.

## 4.2 Prevent IP spoofing

In our threat model we assumed IP spoofing is possible. This assumption is not always true. The Internet community addressed this issue as early as in May 2000 in BCP-38 and suggested that, whenever possible, spoofed traffic should be blocked at the network edge (ingress and

egress filtering). The Open Resolver project suggests in its homepage to configure Source Address Validation / uRPF / BCP-38 on all CPE and Datacenter equipment edges that have fixed IP ranges.

Unfortunately, a significant number of providers still allow IP source address spoofing. At the time of this writing the Spoofer Project indicate that 26% of ASes fully allow spoofing while additional 14.8% at least partially allow it. Geographic Distribution indicates this problem isn't region specific. The Spoofer project is designed to measure the Internet's susceptibility to spoofed source address IP packets by letting volunteers run a program that send a series of spoofed UDP packets to servers distributed throughout the world in order to check if the volunteer's own network allows spoofing. Tough less than 5% of the total number of ASes were tested, up until now it has been the best public resource for this kind of measurement.

**Remote Spoofer Test** was designed and deployed as an alternative to the Spoofer project. This remote test allows identifying thousands of ASes that support IP spoofing from remote without involving any volunteers. The method used relies on DNS forwarders that have bad networking implementation. Normally, a DNS client sends request to DNS resolver who in turns follows the dns tree until it gets the result from an authoritative name server (or a cached result earlier) and sends the result back to the client. Often tough, DNS forwarders (a.k.a. proxies) are used. In this case the forwarder gets the request from the client and forwards it to recursive DNS resolver instead of doing the iterative procedure by itself. There are however badly implemented forwarders. These forwarders fail to replace either the source IP of the request or the source IP of the response. The remote spoofer test takes advantage of these badly implemented forwarders.

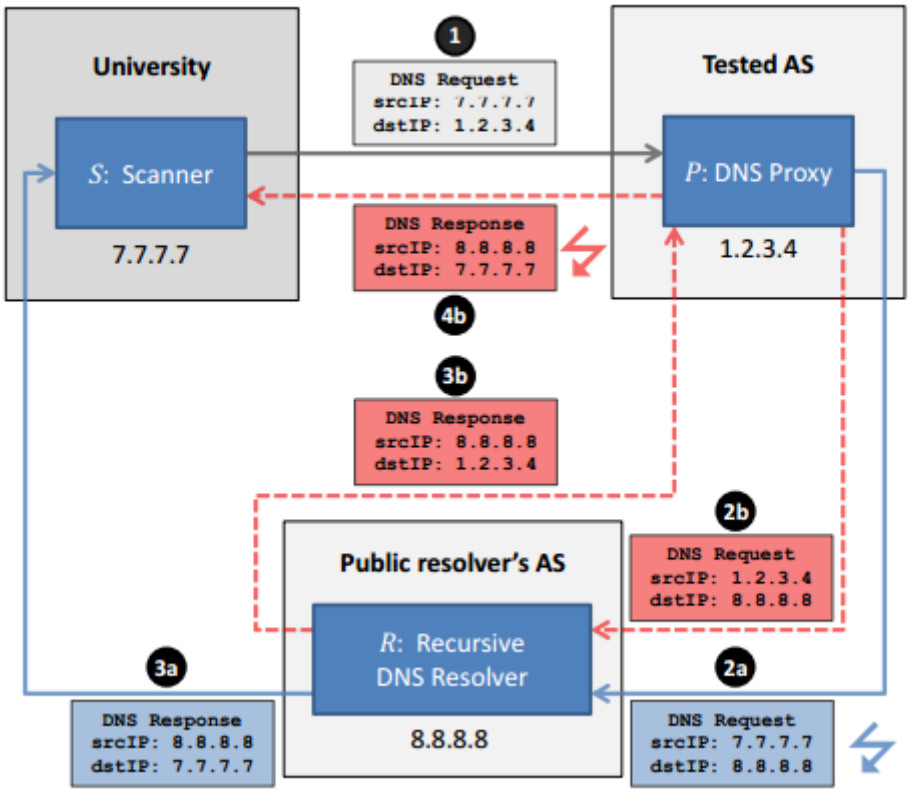


Figure 5: Network overview illustration of the two possible paths for DNS requests and responses when the response received from unexpected source IP address.

To run the remote spoofer test we control a scanner and a name server authoritative for

a specific domain suffix $d_{suf}$. The scanner **S** sends DNS A lookup for domain **d** to an open resolver **P**. The domain d is crafted for each request by prepending a random number (for cache avoidance) and the encoded IP address of P to the suffix $d_{suf}$. At this point we observe the response and match the source IP address to that in the domain d. In many cases there was no match and the response was received from an unexpected source IP address of some resolver we mark as **R**. The reasons for this can vary. It's possible that P is a multi-homed system, but we deny this option for cases where the resolver R is well known (e.g. Google DNS resolver). The other explanation are that P is a proxy with bad network implementation, which can happen due to badly implemented NAT or DNS forwarding software. Figure 5 illustrates the two possible scenarios of getting a response from the wrong source IP in the case of badly configured forwarder. Scenario **a** shows the path when **P** fails to replace the source IP of the request. Scenario **b** shows the path when **P** fails to replace the source IP of the response. When getting a response from unexpected source IP address, if the ASes of S, P and R are different we can conclude that AS(P) allows spoofing. Using this criterion 2,692 ASes were found to allow IP spoofing. It's possible however that P and R reside in the same multi-homed system with interfaces to distinct ASes, so we harden the criterion and only consider resolvers R that respond to more than one forwarder, resulting in 2063 ASes that allow spoofing, and resolvers that responds to more than 10 forwarders, resulting in 870 ASes that allow spoofing. Compare this to the 789 ASes currently identified by the Spoofer Project.

Once IP Spoofing will be eliminated completely the threat of Amplification attacks would go away. It still remains in question whether that would ever happen.

## 4.3 Protocol hardening

Many of the UDP protocols mentioned were not designed with security in mind with respect to amplification considerations. We will discuss possible ways to harden UDP protocols against amplification abuse. We do not consider TCP amplification here as the protocol design is irrelevant in that case.

**Request/Response Symmetry:** The most trivial solution would be designing the protocol in a way that enforces each request to be as big as the response. This trivial solution does indeed prevent amplification but at the cost of lowered effectiveness and higher load in the honest use case. Also note that this solution does not hold for existing protocols due to backward compatibility considerations.

**Rate Limiting:** Enforce a limit on the number of requests per client. In fact, many of the UDP protocol we reviewed do enforce rate limiting, or at least support it. We already mentioned DNS has a rate limiting feature that allows limiting the number of responses per subnet in a given time interval, falling back to truncated messages if the limit is exceeded. Recent Quake 3 implementations only allow one status request per minute from a single IP address. Kad and even the Gameover botnet employ tracking of requests and blacklisting addresses that send too many requests. However, rate limiting does not actually prevent abuse. An attacker can abuse an amplifier in a slow rate but do so for millions of amplifiers resulting in a large aggregated attack volume. An attacker can also spoof multiple IP addresses from the victim's network to avoid host based rate limiting to some extent, so we recommend implementing rate limiting on a per subnet basis. Note however that a bad implementation can allow an attacker to run a DoS attack against the protocol itself and deny access from the victim to the host running the protocol service (by reaching the victim's rate limit on his behalf using ip spoofing). Table 7 shows the results when host based rate limiting of 1 request per ip per second is allowed when the attacker spoof a single victim IP (/32) and when the attacker spoof addresses from the

entire victim subnet (/24). We notice that attacks of up to 22.2Tb/s are still possible.

| 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 7: Aggregated DRDoS bandwidth per protocol for a victim (/32) and a victim's network (/24) if rate limiting is deployed.

| Protocol | (ii) UDP ports | | (iii) Resp len | | (iv) PL | Detection Port | / len | / 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 8: 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

**Session Handling:** A core issue in the vulnerable UDP protocols identified is the lack of session handling. Session handling in a way that enforces the validity of the client identity would prevent the attacker from taking advantage of IP spoofing. Many recent UDP based protocols implement session handling. As an example, in BitTorrent's UDP tracker protocol the server first computes a random connection ID that the client must attach to any further requests. The disadvantages of this solution is the added latency involved due to the connection establishment step, added payload in the form of session identifier to all further requests, and possibly added complexity to the protocol implementation itself. Note that due to backward compatibility considerations, adding session handling is sometimes impossible.

## 4.4 Secure service configuration

Many protocols are suitable for abuse because of bad configuration. Often the default configuration deployed on vendor devices like printers and routers are not good enough and allow amplification abuse. We have seen for both DNS open resolvers and NTP a simple configuration change to only allow authenticated clients access the problematic features can protect against amplification abuse. Once a weak feature is identified in a protocol any network administrator should check if a configuration exists that disables this feature or protect it from being abused.

## 4.5 Packet filtering solutions

In the NTP case study we observed that packet filtering was used to identify and discard monlist requests. Reactive countermeasures are indeed being used as a last resort against DDoS attacks. Typically, four packet based filtering techniques are used to mitigate DDoS attacks: (1) IP address filters (2) UDP and TCP ports filters (3) packet length filters (4) payload string matching. The first is irrelevant for DRDoS attacks. Table 8 summarizes the results of using of the last three packet filtering techniques to detect attack traffic. It actually doesn't separate attack from legitimate traffic, but can be used as a first step to pre-select candidates for later inspection by more sophisticated tools. You can notice that for 7 protocols a port match catch

95% of packets. Length based filtering seems ineffective. Payload matching can be used for 10 of the protocols which have static substrings of at least 7 bytes. Note that this is only a first step, but can be used as a truly last resort at the cost of high false positive rate.

# 5 Discussion and Future work

We've seen how some UDP protocols and TCP handshake can be abused for DRDoS attacks. The US-CERT issued an Alert (TA14-017A) for UDP-based Amplification Attacks listing the 14 UDP protocols identified as potential attack vectors for DRDoS attacks. It seems like there is still no discussion on the potential threat of TCP amplification attacks.

For TCP we saw unexpected SYN responses that allow abuse. It remains question if some UDP protocols that employ a handshake mechanism can be abused like TCP.

We've seen that preventing IP Spoofing can prevent reflection attacks. It remains in question whether IP Spoofing can be avoided completely. The fact that many networks still allow IP Spoofing indicates ISPs have little intensivity towards implementing spoofing prevention. Why? What can be done to force it?

We mentioned countermeasures against DRDoS attacks. We did not list popular methods used to prevent DDoS attacks in general such as global anycast cloud networks that spread the attack volume on multiple networks, solutions like Cloud-based Elastic Relay Network (WIP by A. Herzberg) and more.

An interesting future attack is combining Crossfire [3] and Amplification. This would allow higher impact at lower cost making this attack much more feasible in real life. Some key changes must be made to the Crossfire attack to allow use of amplifiers. E.g. the algorithm for target links selection must be changed as an attacker normally don't have access to the amplifiers themselves and cannot run traceroute from amplifier to the target area servers and decoy servers.

It would also be interesting to investigate related attacks based not on reflection but on 'puppets' (scripts running in sandbox) and in particular 'persistent puppets' (scripts added to an object loaded by some page often used by user).

# References


[1] Hupperich T. Rossow C. Kuhrer, M. and T Holz. Exit from hell? reducing the impact of amplification ddos attacks.

[2] Hupperich T. Rossow C. Kuhrer, M. and T Holz. Hell of a handshake: Abusing tcp for reflective amplification ddos attacks. *USENIX Workshop on Offensive Technologies (WOOT).*

[3] S. B. Lee M. S. Kang and V. D. Gligor. The crossfire attack. *Proceedings of IEEE Security and Privacy, San Francisco, CA, USA.*

[4] Christian Rossow. Amplification hell: Revisiting network protocols for ddos abuse. *USENIX Workshop on Offensive Technologies (WOOT).*

[5] A. Studer and A. Perrig. The coremelt attack. *Proceedings of the European Symposium on Research in Computer Security (ESORICS), Saint Malo, France.*