Exit from Hell? Reducing the Impact of Amplification DDoS Attacks

Marc Kührer, Thomas Hupperich, Christian Rossow, Thorsten Holz
Horst Görtz Institute for IT-Security, Ruhr-University Bochum

Abstract

Amplification vulnerabilities in many UDP-based network protocols have been abused by miscreants to launch Distributed Denial-of-Service (DDoS) attacks that exceed hundreds of Gbps in traffic volume. However, up to now little is known about the nature of the amplification sources and about countermeasures one can take to remediate these vulnerable systems. Is there any hope in mitigating the amplification problem?

In this paper, we aim to answer this question and tackle the problem from four different angles. In a first step, we monitored and classified amplification sources, showing that amplifiers have a high diversity in terms of operating systems and architectures. Based on these results, we then collaborated with the security community in a large-scale campaign to reduce the number of vulnerable NTP servers by more than 92%. To assess possible next steps of attackers, we evaluate amplification vulnerabilities in the TCP handshake and show that attackers can abuse millions of hosts to achieve 20x amplification. Lastly, we analyze the root cause for amplification attacks: networks that allow IP address spoofing. We deploy a method to identify spoofing-enabled networks *from remote* and reveal up to 2,692 Autonomous Systems that lack egress filtering.

1 Introduction

Distributed Denial-of-Service (DDoS) attacks have been known since many years [8,9,24,34] and they still constitute an important problem today. For a long time, DDoS attacks were hard to tackle due to their *semantic* nature: it is difficult to distinguish an actual attack from a sudden rise in popularity for a given service due to a flash crowd ("Slashdot effect"). A large body of literature is available on this topic and many DDoS detection mechanisms and countermeasures have been proposed over the years (e.g., [14,15,43]). Furthermore, advances in *Cloud computing* and load balancing techniques helped to mitigate this problem [17,18], and nowadays simple types of DDoS attacks such as *SYN* and *UDP flooding* are well-understood.

However, the adversaries evolved and modern DDoS attacks typically employ so called *amplification attacks*, in which attackers abuse UDP-based network protocols to launch DDoS attacks that exceed hundreds of Gbps in traffic volume [21, 22]. This is achieved via *reflective* DDoS attacks [31] where an attacker does not directly send traffic to the victim, but sends spoofed network packets to a large number of systems that reflect the traffic to the victim (so called *reflectors*). Often, attackers choose reflectors that send back responses that are significantly larger than the requests, leading to an increased (*amplified*) attack volume. We call such reflectors *amplifiers*. Recently, many types of such amplification attacks were discovered [33]. However, little is known about the nature of the amplifiers and about countermeasures one can take to remediate vulnerable systems.

In this paper, we address this problem and study the root causes behind amplification DDoS attacks. We tackle the problem from four different angles and provide empirical measurement results based on Internet-scale scanning to quantify the problem.

In a first step, we want to understand the nature of amplifiers and determine which kinds of systems are vulnerable. Previous work on empirically understanding DDoS attacks typically focused on ways to estimate the size of the problem and understanding the infrastructure behind such attacks [1, 5, 26]. To increase the understanding of amplification attacks, we utilized protocol-specific fingerprinting to reveal as much information as possible from systems that can be abused on the Internet. More specifically, we enumerated the amplifier sources for seven network protocols and performed large-scale scans to collect information about vulnerable systems. This enables us to categorize the types of devices that can be abused in the wild. We found that there is a large diversity of vulnerable devices and analyzed their properties. For example, we found 40.8% of the vulnerable NTP hosts to run Cisco IOS, an OS that is deployed on Cisco network devices.

Based on these insights on amplifiers, a viable next step is to reduce the number of vulnerable systems on the Internet. Previous work on that topic mainly focused on understanding botnet Command & Control servers that are used to orchestrate classical DDoS attacks [5]. However, modern amplification attacks use a completely different modus operandi. We contributed to a global security notification procedure where our scanning results were used to notify NOCs and CERTs of hundreds of large ISPs worldwide about NTP servers vulnerable to amplification attacks. Furthermore, we collaborated with security organizations in order to create advisories that describe the technical background and approaches to solve the problem. We analyzed the remediation success of these measures and found that the number of NTP servers vulnerable to `monlist` amplification dropped by 92% in a 13-week period between November 2013 and February 2014. We closely analyzed this effect and found that especially vulnerable NTP servers within ARIN have been mitigated, while other geographic regions lag behind.

Since it seems feasible to significantly reduce the number of amplifiers, a third angle of the problem is an analysis of potential attack vectors that adversaries could abuse in the future. We start with the basic insight that up to now UDP-based protocols are leveraged by attackers, since these protocols provide large amplification factors. We study a completely different kind of amplification attacks, namely TCP-based ones. Surprisingly, even TCP can be abused for amplification attacks, despite the fact that this protocol uses a 3-way handshake. This is due to the fact that certain TCP stacks retransmit `SYN/ACK` packets multiple times (some 20x or more) when they presume that the initial `SYN/ACK` segment was lost. Thus an amplification of 20x or more is possible. Empirical scan results suggest that there are hundreds of thousands of systems on the Internet that can be abused this way. We performed protocol-specific fingerprinting to learn more about the nature of such devices.

As a fourth angle of the problem, we analyzed the root cause behind amplification attacks: if a given network does not perform *egress filtering* (i.e., verifies that the source IP address in all outbound packets is within the range of allocated internal address blocks, see BCP 38 [13] for details), an attacker can spoof packets and thus initiate the first step of reflective DDoS attacks. Identifying such networks is a challenging problem [12] and existing solutions rely on a client deployed in the network under test [3, 36]. We utilize a novel remote test based on DNS proxies that enables us to identify thousands of Autonomous Systems that support IP spoofing. To summarize, our contributions are as follows:

- We performed Internet-wide scans to identify and monitor all relevant potential amplifiers for seven network protocols vulnerable to amplification attacks. We fingerprint and categorize these systems, showing a high diversity in the amplifier landscape.
- We study the success of a global security notification campaign to alert administrators of vulnerable NTP servers and show the benefits and limitations of such large-scale initiatives.
- Aiming to assess further amplification DDoS techniques, we identify TCP as an alternative source for amplification—despite its three-way-handshake protocol. We reveal millions of systems that can be abused to amplify TCP traffic by a factor up to 20x.
- Finally, we aim to tackle the root cause for amplification DDoS attacks: networks that do not perform egress filtering and thus allow IP address spoofing. We deploy a *remote* scanning technique and find up to 2,692 ASes that permit spoofed IP traffic.

Paper Outline. The paper is organized as follows. In Section 2, we define the threat model and outline our scanning setup to perform Internet-wide scans. We then shed light onto the landscape of hosts that are vulnerable to UDP-based amplification DDoS attacks. In Section 3, we detail the effects of our NTP case study. Section 4 tackles the problem of TCP-based amplifiers, demonstrating that the TCP three-way-handshake can be abused for amplification attacks. In Section 5, we introduce a novel mechanism to identify networks that allow IP address spoofing. Section 6 reviews prior work and we conclude this paper in Section 7.

2 Amplification DDoS

We begin with an analysis of the threat landscape. To this end, we first review the general threat model before we analyze different aspects of amplification DDoS attacks. More specifically, we study the amplifier magnitude, measure what kinds of devices can be abused on the Internet, and determine the churn of amplifiers.

2.1 Threat Model

The scope of this work are amplification DDoS attacks. In such an attack, a miscreant abuses public systems (such as open recursive DNS resolvers) to reflect attack traffic to a DDoS victim [31]. In particular, she abuses hosts that not only reflect but also amplify the traffic. Typically, the attacker chooses connection-less protocols in which she can send relatively small requests that result in significantly larger responses. By spoofing the source of the traffic (i.e., impersonating the victim), she can enforce that the public systems—unwillingly—amplify and reflect traffic to the victim. Prior work has revealed that at least 14 UDP-based protocols are vulnerable to such abuse [33]. These protocols offer severe amplification rates—in the worst case, as with the `monlist` feature in NTP, they amplify traffic by a factor of up to 4,670.

2.2 Amplifier Magnitude

In this paper, we try to shed light onto the landscape of *amplifiers*, i.e., hosts that are vulnerable to amplification abuse. As a first step, we enumerate and observe these amplifiers in the IPv4 address space. That is, we performed Internet-wide scans for a subset of the vulnerable protocols: DNS, SNMP, SSDP, CharGen, QOTD, NTP, and NetBIOS. We chose to monitor these protocols, as prior work only approximated the amplifier landscape for them. The amplification vulnerabilities of these seven protocols can be abused by attackers to launch severe amplification attacks. In addition, all these seven protocols run server-side, thus hosts running such protocols are seemingly better connected and more stable in terms of IP address churn than hosts of end users.

Scanning Setup. We developed an efficient scanner to identify amplifiers for these protocols in Internet-wide scans. In order to respect good scanning practices as suggested by Durumeric et al. [11], we limit the number of requests that a particular network receives. For this reason, we compute the scan targets as a pseudo-random permutation of the entire IPv4 address space (except the IP address 0.0.0.0). That is, we use a linear feedback shift register (LFSR) to compute the order of the $2^{32}-1$ IPv4 addresses to be scanned. In order to avoid to become blacklisted, we refrained from aggressive scanning and distributed the scans over 48 hours. In addition, we set up a reverse DNS (rDNS) record for our scanner and configured a web server that presents project information and an explanation how to opt-out from our scans.

For each of the protocols, we send a request that can be used to amplify traffic. That is, we send NTP `version` requests, SSDP `SEARCH` requests, SNMP v2 `GetBulk` requests, DNS `A` lookups, and NetBIOS' default name lookup. We ran the scans on a weekly basis from Nov 22, 2013 to Feb 21, 2014 to observe potential changes in terms of amplifiers. We chose to use the weekends for our scans so that the load of both our scanning network and the scanned networks have less impact on business activities. In the case of CharGen and QOTD, we refrained from repeating the scans, as the number of amplifiers was too low to justify repeated full scans.

During the course of our scans, we received 90 emails from administrators asking about the scanning experiments. Adhering to these requests, we excluded 91 IP prefixes and 30 individual IP addresses (about 3.7 million IP addresses in total) after administrators asked us to do so. To allow comparisons between two scans, we ignored these IP addresses in all of our scans, i.e., even if they were not blacklisted at the beginning.

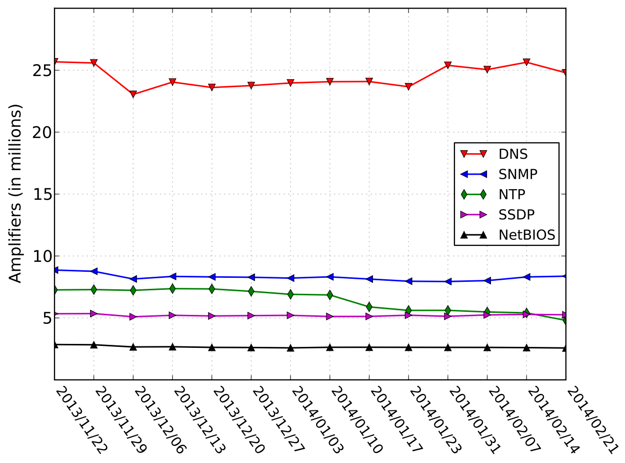

Figure 1: Trend of UDP-based amplifiers

Table 1: Intersection of potential amplifiers based on the Internet-wide scan on Nov 22, 2013

	Intersection (in %)						
Protocol	*DNS*	*CharGen*	*NetBIOS*	*NTP*	*QOTD*	*SNMP*	*SSDP*
DNS	-	0.0	0.5	0.2	0.0	11.6	2.1
CharGen	1.7	-	2.4	20.0	4.0	9.4	1.3
NetBIOS	4.7	0.1	-	0.6	0.2	1.8	5.9
NTP	0.9	0.3	0.2	-	0.0	3.2	0.1
QOTD	14.0	11.8	18.5	8.4	-	4.2	8.5
SNMP	33.5	0.1	0.6	2.7	0.0	-	0.2
SSDP	9.9	0.0	3.1	0.2	0.1	0.4	-

Results. Figure 1 illustrates the number of identified amplifiers per protocol. By far the highest number of amplifiers was found for open recursive DNS resolvers, slightly fluctuating between 23 and 25.5 million systems. For most of the other protocols, the number of amplifiers is quite constant. An exception are NTP amplifiers, whose popularity constantly decreases, a phenomenon that we describe in detail in Section 3. None of the protocols (except for the two legacy protocols CharGen with 107,725 and QOTD with 36,609 vulnerable hosts) had fewer than 2.5 million amplifiers, showing a large landscape of hosts that can be abused.

Quite interestingly, some systems run multiple vulnerable services. Table 1 shows the intersection between the individual protocols relative to the overall number of amplifiers for the protocols specified in the first table column. The largest overlap is between DNS and SNMP: a third of the public SNMP hosts also run open recursive DNS resolvers. Note that the table is not symmetric, which, for example, reveals that less than 11.6% of the open DNS resolvers also run unprotected SNMP daemons. For most of the other protocols the intersection is negligible, though. This means that the number of amplifiers basically sums up. We measured almost 46 million amplifiers for all scanned UDP-based protocols.

Table 2: Results of the device fingerprinting for the amplifiers identified on Nov 22, 2013

	Hardware (in %)				Architecture (in %)					Operating System (in %)										
Protocol	*Router*	*Embedded*	*Others*	*Unknown*	*x86*	*MIPS*	*PowerPC*	*Others*	*Unknown*	*Unix*	*Linux*	*Ubuntu*	*FreeBSD*	*Windows*	*ZyNOS*	*Cisco IOS*	*Junos*	*NetOS*	*Others*	*Unknown*
DNS	9.7	5.7	0.6	84.0	0.6	7.0	0.0	0.4	92.0	3.6	3.4	0.0	0.0	0.8	7.5	0.1	0.0	0.0	1.1	83.5
NetBIOS	0.7	1.3	2.0	96.0	87.6	0.1	0.0	0.0	12.3	0.4	0.1	0.0	0.0	87.3	0.3	0.0	0.0	0.0	0.7	11.2
NTP	44.8	0.5	2.4	52.3	9.6	18.4	6.9	1.1	64.0	18.2	26.8	0.0	4.7	0.2	0.0	40.8	2.9	0.0	1.7	4.7
SNMP	66.5	10.4	3.1	20.0	2.9	44.9	1.1	3.1	48.0	1.5	11.4	0.1	0.1	0.8	17.8	2.2	0.0	0.0	8.7	57.4
SSDP	94.3	2.9	2.2	0.6	1.5	2.7	0.0	0.1	95.7	1.8	36.0	5.5	0.0	1.3	0.7	0.0	0.0	19.3	1.8	33.6

2.3 Amplifier Classification

Observing the magnitude of the problem, we wondered what kinds of systems allow for such amplification vectors. In an attempt to answer this question, we use protocol-specific fingerprinting to reveal as much information as possible from these systems. That is, we generate device fingerprints by inspecting the replies from the amplifiers during our UDP scans. We dissect the responses of each host and protocol individually to classify systems in three categories: the underlying hardware (e.g., routers, desktop computers, or printers), the system architecture (such as x86, MIPS, or PowerPC), and the operating system.

Fingerprinting Setup. We manually compiled 1,873 regular expressions that allow a fine-granular generation of fingerprints. We further leverage Nmap service probes [27] to fingerprint the NetBIOS protocol. For NetBIOS, we also focus on the structure of the payload to obtain information about the OS. NTP `version` responses reveal the processor type, OS, and the version of the running NTP daemon. To generate fingerprints for the SNMP protocol, we analyze the object identifier values (OID) in the responses. For SSDP, the responses contain text fragments resembling HTTP headers that provide system information in the `Server` header field. Additionally, SSDP headers include Unique Service Name (USN) and Search Target (ST) fields, providing more general information about a device.

We improve the coverage of our UDP fingerprints by scanning the amplifiers for common TCP-based protocols. We use FTP, HTTP, HTTPS, SSH, and Telnet to leverage information in protocol banners and text fragments. We synchronize the TCP and UDP scans, hence once a full Internet-wide UDP scan is finished, we initiate a follow-up TCP scan for hosts that are found to be an amplifier for at least one UDP protocol.

Results. Table 2 depicts the fingerprint results obtained for the amplifiers found on Nov 22, 2013. For reasons of brevity, we summarize fingerprint details with less than 2% share to *Others*. Note that the category "router" also includes gateways, switches, and modems as many of these devices provide similar features.

The best results for the OS classification are achieved for NTP. We find that 40.8% of the vulnerable NTP hosts run Cisco IOS, an OS that is deployed on Cisco devices such as business routers and switches. We further identify 1,267,008 amplifiers (17.4%) running Linux on MIPS and 357,076 devices (4.9%) running Linux on PowerPC. These two combinations are common for consumer devices such as routers and modems. The majority of NTP amplifiers thus run on networking equipment.

Similarly, two thirds of the SNMP amplifiers are routers. With a share of 17.8%, the ZyNOS system stands out—apparently running unprotected SNMP services per default. But we also observe a wide distribution of other SNMP devices. This includes 58,000 office printers (0.7%), 51,037 firewall appliances (0.6%), and 40,061 network cameras (0.5%). Routers are even more prominent among SSDP hosts with a share of about 94.3%. This shows that at least three of the analyzed protocols are overly prominent on routers.

On the contrary, the vast majority of NetBIOS amplifiers run Windows on x86, a typical setup of desktop computers. Since the Conficker outbreak in 2008, it is known that millions of Windows systems on the Internet are reachable via the NetBIOS [32] protocol.

Unfortunately, DNS provides only limited fingerprint information and we thus had to solely rely on the TCP fingerprints to classify DNS servers. However, most DNS servers did not run TCP services, resulting in a high number of uncategorized hosts. Even if TCP services were accessible, the provided information was often too generic (e.g., banners as "Apache", "SSH-2.0-OpenSSH", or "FTP Server"). However, we could identify 5.4% of the hosts (1,388,348) as MIPS-based routers with ZyNOS, which is common for broadband routers distributed by manufacturer ZyXEL.

A high diversity of amplifiers is attested when looking at smaller clusters. For example, we find 695 vulnerable devices to be running *Miele Logic*, a payment system for Miele devices such as washing machines. Similarly, we identify 9,224 amplifiers running server man-

Table 3: Amplifier churn rate per protocol

	Initial Scan	Week 1		Week 13	
Protocol	(#)	(#)	(%)	(#)	(%)
DNS	25,681,450	12,190,302	47.5	8,263,508	32.2
NetBios	2,853,213	1,455,351	51.0	979,266	34.3
NTP	7,269,015	6,859,043	94.4	4,222,060	58.1
SNMP	8,866,748	4,939,118	55.7	3,411,563	38.5
SSDP	5,336,107	3,088,148	57.9	2,067,830	38.8

agement systems (like iLO, iDRAC, or IPMI). We further find 51,351 Digital Video Recorders, 7,739 Power Distribution Units, and 20,927 Network Attached Storage devices (NAS) to be vulnerable to amplification.

Ambiguous Fingerprints. We had to resolve a few conflicts when combining the fingerprints from multiple protocols. For NTP amplifiers, we find valid TCP fingerprints for 1,919,932 hosts, while conflicts emerge for 9,945 IP addresses (0.5%). For SNMP, we leverage TCP data for 2,042,541 amplifiers while obtaining 31,346 conflicts (1.5%). We presume that these conflicts were caused by responses from "border" devices such as routers that host some services themselves (e.g., SNMP and SSH), while requests for FTP or HTTP were forwarded to the devices connected to the router, resulting in multiple fingerprints for a single IP address. To resolve these conflicts, we assign a lower priority to TCP fingerprints when classifying the amplifiers. In addition, we refrained from aggregating the individual UDP fingerprints to one large set, as the overlap between the UDP protocols is low anyway (cf. Table 1).

2.4 Amplifier Churn

An important aspect from the attacker's point of view is how fast the set of amplifiers changes. An up-to-date list of reliable amplifiers is key to achieving a high impact during an attack. For this reason, we measure the *churn rate* of the amplifiers per protocol, which shows how fast a list of amplifiers becomes outdated. That is, we enumerate the amplifiers based on their IP addresses on Nov 22, 2013 and check if these hosts are still vulnerable for amplification attacks in the subsequent weeks.

Table 3 lists the numbers of amplifiers for the five UDP protocols that we monitored on long term. Figure 2 illustrates the ratio of amplifiers that are still reachable at the same IP address over time. For most protocols (DNS, NetBIOS, SNMP, and SSDP) the churn of amplifiers is quite high: only about 50% of the initial hosts are still reachable after one week. After the second week, we again observe a minor decrease, resulting in a total of 40 - 50% of available amplifiers for each protocol. For the following weeks we find the number of amplifiers to reach an almost steady level.

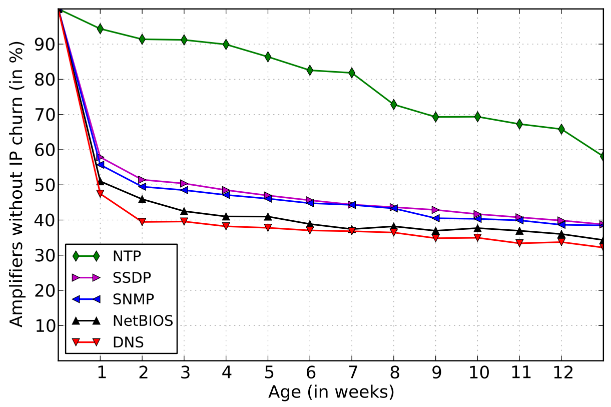

Figure 2: IP churn of potential amplifiers

To understand the nature of the significant drop after the first week and obtain knowledge about the vulnerable systems still reachable after 13 weeks, we leverage our fingerprinting techniques. We find the amplifiers that became outdated within the first week to be mostly connected via consumer routers (e.g., 78.8% for SNMP). We assume that these routers are connected via DSL with low IP address lease times, causing the rapid breakdown rate after one week [37]. To confirm this assumption, we aggregated reverse DNS records for a random sample of 100,000 unreachable amplifiers and checked for common indicators of dial-up connections (i.e., the appearance of tokens such as "dialup", "dyn", or "pool"—we further manually verified the Top 5 providers not providing indicators in the rDNS data). We indeed found at least 82.8% of the IP addresses to be linked to connections with dynamic IP address assignment. This means that an attacker needs to frequently re-scan for amplifiers or otherwise risks to decrease the impact of her attacks.

The amplifiers that are still reachable after 13 weeks presumably have longer lease times or static Internet connectivity. For example, we can see a clear distinction between countries in which SSDP hosts disappeared after a week (e.g., China, Argentina, Russia) and countries in which most hosts are still reachable after 13 weeks (e.g., Korea, United States, Canada). While only 3.4% of the Chinese amplifiers were still reachable after 13 weeks, still more than 69% of the Canadian amplifiers were available. This shows that the geolocation of Internet links (and thus the risk to face IP address churn) highly influences the availability of amplifiers.

Interestingly, the NTP protocol draws a completely different picture. Given a fixed list of vulnerable hosts, the ratio of available NTP amplifiers decreases at a negligible rate. After four weeks, an attacker can still abuse approximately 90% of the initial NTP amplifiers. After 13 weeks, still 58.1% of the initially-scanned hosts are reachable. Of the 41.9% decrease after 13 weeks, many systems presumable disappeared because of our NTP amplifier notification campaign (cf. Section 3)—and not because of IP churn. The actual churn is thus even lower and significantly differs from churn in the other protocols. In contrast to the vulnerable amplifiers of the other protocols, more than 40% of the NTP amplifiers are running Cisco IOS that is commonly distributed on business routers and switches with static IP addresses. We further find 53.7% of the vulnerable hosts to be located within the United States (31.3%), South Korea (13.0%), and Japan (9.4%) for which the typical IP lease times for broadband are above average.

3 Case Study: NTP Amplification

After inspecting the amplification attacks in general, this section focuses on NTP, which we consider by far the worst among all known vulnerable protocols. NTP is a promising amplification vector for an attacker for three reasons. First, NTP server implementations allow for amplification factors of up to 4,670 [33]. Attackers can abuse the `monlist` feature in popular `ntpd` versions, which requests a list of up to 600 NTP server clients in about 44kB UDP payload. Second, as we have seen in Section 2.4, NTP servers have minimal IP address churn. Lastly, NTP offers even further amplification vectors. For example, the NTP `version` request reveals a verbose system fingerprint (OS, architecture, server info) of the NTP server, allowing about 24-fold amplification.

Attackers have already practically demonstrated the impact of NTP attacks. For example, in February 2014, CloudFlare observed a 400 Gbps attack against a French hosting provider [22]—the largest DDoS attack observed so far. If the attacker had even more resources (in particular bandwidth) to send spoofed `monlist` requests, the impact of such an attack could have been even higher.

Luckily, NTP servers can be configured such that the `monlist` requests are disabled for unauthorized users, and more recent `ntpd` versions protect this feature with a proper session handshake. These changes typically do not bring disadvantages for the administrators, while they eliminate the amplification vector. Even disabling functionality like `monlist` does not break time synchronization. But although secure configurations are well-documented, most administrators are not aware of the amplification vulnerabilities and operate NTP servers in (sometimes bad) default configurations. From a security-perspective this raises several urging questions: once we found amplification vulnerabilities, how can we reduce the number of amplifiers? Can we notify administrators? How effective would such a notification procedure be?

3.1 NTP Amplifier Notifications

In a large-scale campaign, we have launched a global notification procedure to alert NTP administrators about the amplification problems. We thankfully cooperated with many parties striving towards the same goal: reducing the number of NTP amplifiers.

Datasets. We define two datasets of NTP amplifiers. NTP_{ver} contains all NTP servers that reply to `version` requests, i.e., systems that are “less” vulnerable to amplification abuse. This is the same dataset that we fingerprinted in Section 2. As a subset of this, NTP_{mon} contains the NTP servers that also support the `monlist` requests, i.e., systems that allow for more “severe” amplification.

Campaign. We collaborated with security organizations in order to create technical advisories that describe how to solve the amplification problems in NTP. This resulted in public advisories of CERT-CC [42] and MITRE [25]. Due to the high number of vulnerable Cisco devices for NTP amplification (cf. Table 2), we also contacted Cisco which resulted in a public advisory of Cisco’s PSIRT [7]. The advisories describe how to disable the `monlist` feature in typical NTP server implementations (such as `ntpd`). The same configuration change also disables `version` responses. Thus, in principle, the advisories help to reduce the number of servers in both datasets, NTP_{ver} and NTP_{mon}.

In addition, we distributed lists of IP addresses of the systems in NTP_{mon} among trusted institutions. For example, we shared our data with direct contacts in NOCs and CERTs of hundreds of large ISPs worldwide. Furthermore, we cooperated with data clearing houses (e.g., TrustedIntroducer [41] and ShadowServer [35]) that informed their subscribers. Lastly, we informed the NTP Pool Project [28] about misconfigured hosts in the public pool of NTP servers and synchronized our notifications with the OpenNTPProject [30] to start the announcements simultaneously.

We did not actively notify systems that are only in NTP_{ver} for two reasons. First, we saw an urgent need to close the amplifiers in NTP_{mon}, as the `monlist` amplification is in the order of magnitudes higher than of other NTP features. Second, we can then compare the efficiency of advisories (which affect both datasets) with the effects of active and personalized notifications (which affect only NTP_{mon}).

3.2 Analyzing the Remediation Success

We ran weekly scans for NTP amplifiers to observe the developments over time. Figure 3 shows the number of NTP amplifiers per week and marks important events.

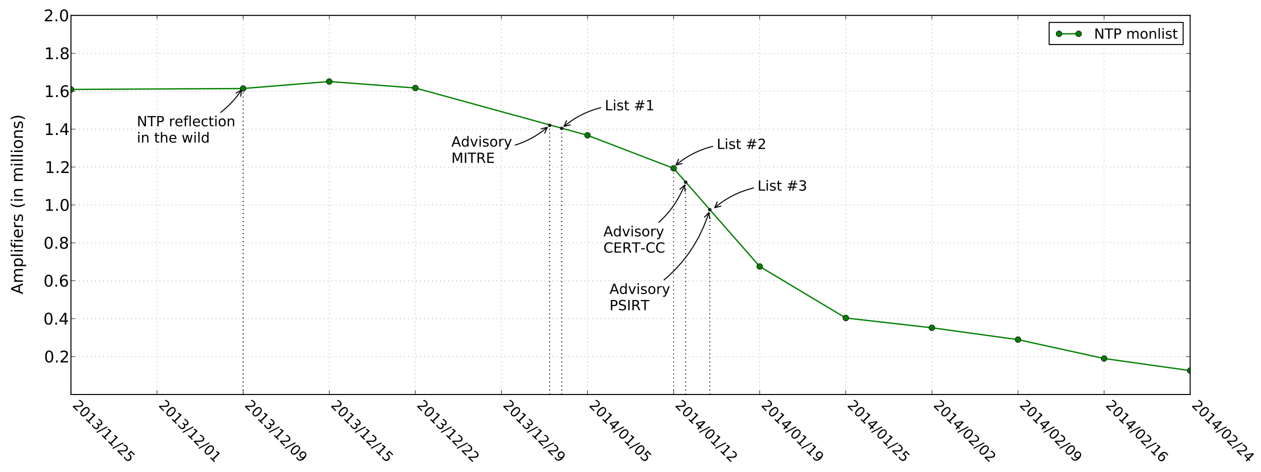

Figure 3: Trend of NTP_{mon} amplifiers

At their peaks on Dec 15, 2013, we tracked 7,364,792 servers in NTP_{ver} and 1,651,199 servers in NTP_{mon}, respectively. The number of amplifiers in NTP_{mon} is steady at first, also after Symantec has released a blog article about the attacks late December 2013 [39]. However, the number of amplifiers starts to drop (15.4%) right after we released the CVE and shared a first (incomplete) list of IP addresses. A second major drop (43.4%) happened during the week we distributed twice an updated (and complete) list of potential amplifiers and two other advisories were released. After publishing further weekly notifications (omitted from the graph for better readability) we have observed a steady decrease of amplifiers.

At the end of our measurements on Feb 24, 2014, the number of amplifiers reached a local minimum at 4,802,212 (NTP_{ver}) and 126,080 (NTP_{mon}). Compared to the peak numbers, this constitutes significant drops in both datasets. The number of amplifiers in NTP_{ver} decreased by 33.9%, a success likely stemming from the advisories and recent publicity on NTP attacks in general. However, looking at the development of *severe* amplifiers shows how successful global notification efforts can be: the number of amplifiers (NTP_{mon}) dropped by 92.4% with an ongoing decrease.

To verify whether the number of amplifiers for NTP_{mon} still decreases continuously, we performed another Internet-wide scan on Jun 20, 2014 and find 87,463 hosts still vulnerable to `monlist` amplification, i.e., a decrease of almost 40,000 hosts since Feb 24, 2014.

Fingerprinting. We compared the fingerprints of the NTP_{mon} datasets at the start and end of our measurements. We clearly observe decreasing numbers for all architectures, OSes, and hardware types. Interestingly, the ratio of MIPS-based amplifiers dropped from 47.2% to 19.1%, while the ratio of x86-based systems increased from 40.2% to 58.0%. Similarly, 23.0% of the devices of a popular router manufacturer remain vulnerable—a value standing out from the average decrease. On absolute scale, though, the numbers drop across all fingerprints, indicating that the clean-up was not driven only by a single device type or manufacturer.

Table 4: Decrease of NTP_{mon} amplifiers per country

	Amplifiers (in #)			Remaining
Country	Nov 22, 2013	Feb 24, 2014	Decrease	(in %)
US	1,073,666	28,415	1,045,251	2.6
KR	88,289	16,183	72,106	18.3
RU	58,519	11,476	47,043	19.6
DE	50,627	4,793	45,834	9.5
CA	36,070	1,881	34,189	5.2
CN	32,995	4,172	28,823	12.6
JP	29,915	2,777	27,138	9.3
GB	24,408	2,741	21,667	11.2
UA	19,270	2,716	16,554	14.1
BR	13,900	2,719	11,181	19.6
TW	13,362	6,397	6,965	47.9
NL	13,122	3,934	9,188	30.0
FR	12,992	4,557	8,435	35.1
CZ	11,825	1,226	10,599	10.4
PL	10,891	1,960	8,931	18.0

Geographic Distribution. We also investigated the geographical distribution of the amplifiers. For this, we used the MaxMind GeoIP database [23] to assign a country to the IP address of an amplifier. We then compared how the numbers of amplifiers evolve in single countries. Table 4 lists the remaining amplifiers of the 15 countries, which had the most amplifiers in Nov 2013. The clean-up was—on relative scale—most successful in the US, where the number of amplifiers decreased to only 2.6%. In other countries like Taiwan the number decreased only to 47.9%. These differences may be caused by the number and quality of direct contacts we had in the US compared to Taiwan: we admittedly had more contacts in the

Table 5: Decrease of NTP_{mon} amplifiers per RIR

	Amplifiers (in #)			Remaining
RIR	Nov 22, 2014	Feb 24, 2014	Decrease	(in %)
ARIN	1,112,422	30,766	1,081,656	2.8
RIPE	283,991	53,324	230,667	18.8
APNIC	202,719	38,122	164,597	18.8
LACNIC	21,721	5,075	16,646	23.4
AFRINIC	7,495	920	6,575	12.3

US and Europe compared to the rest of the world. But it also shows that the current network of CERTs is not perfectly connected to share our information equally in all countries. For example, also European countries like France (35.1%) and Austria (47.1%) lag behind the average decrease. However, on an absolute scale, the situation is different. While the number of NTP amplifiers was clearly reduced in the US, still 28,415 systems remain vulnerable to `monlist` amplification.

Table 5 shows the absolute numbers of NTP servers per Regional Internet Registry (RIR), which (very roughly) indicates the continent of the amplifiers. It shows that we face a global problem, but also proves that all regions in the world have acknowledged the problems.

Per-Provider Statistics. A closer look at the Autonomous Systems (AS) distribution sheds light onto how amplifiers have been closed. Of the 96 ASes that had at least 1,000 amplifiers, about half have shut down more than 95% of the amplifiers. This shows that many providers either enforce most amplifiers in their networks to be shut down or successfully filter NTP traffic at the network level. More specifically, we identified 73 ASes (0.44% of all ASes we observed during our monitoring period) that had more than 100 NTP servers in one week, and did not have a single vulnerable server left open in the subsequent weeks. This strongly suggests that these providers perform network-level filters, as it is unlikely that so many individual servers were all cleaned up within a few days. Nine ASes left open more than half of the amplifiers, i.e., these providers do very little to mitigate the threat. We are currently establishing individual contacts with the least-active ASes and hope to understand the reasons for the remaining amplifier landscape in their networks.

Result Verification. We verified if the drop in NTP amplifiers is not caused by networks blocking our scanner [10]. This can already be seen in the amplifier trend graph (Figure 1), in which the number of amplifiers for other protocols remains almost constant. To be sure, we scanned for NTP amplifiers from a secondary host in a different `/16` network. The primary scanner indeed missed 8.6% of the amplifiers that the secondary scanner found. We manually investigated this and found 904 networks (`/20`) that have at least five amplifiers that the primary scanner missed—indicating that some networks do blacklist our scanner. While our primary scanner has thus missed amplifiers, these systems make up an almost-negligible part of the 92.2% decrease of amplifiers.

3.3 Lessons Learned

Summarizing, the campaign to reduce the number of NTP `monlist` amplifiers was quite effective and showed remediation successes for almost 95% of the vulnerable hosts just after 6 months. As such, it would be interesting to see recipes to repeat this success in similar campaigns for other security-critical issues, such as amplification vulnerabilities in other protocols or even unrelated, but security-critical problems like the `heartbleed` vulnerability in OpenSSL. Figure 3 clearly shows that the counteractions (advisories and IP address lists) correlate with the decrease in numbers of amplifiers.

Unfortunately, it is impossible to proof causality, in particular, to see which IP address distribution channels or which advisories were most effective. However, in our conversations with providers we had the impression that it helps to repeatedly point out the problem. Further, it may not be sufficiently effective to have public advisories that nobody reads. Instead, we found that communication was key to motivate CERTs and providers to act accordingly. Once we reached out to CERTs and providers, it typically was no problem to close the vulnerable hosts.

On the negative side, though, we experienced that the Internet community is not well-prepared for such campaigns. Although we were quite well-connected with nationally and internationally operating CERTs and providers, it is hard to reach out to all providers individually. If providers and CERTs were better connected to non-profit data clearing houses (like `shadowserver.org`), vulnerability notifications could be sent out more efficiently.

4 TCP-based Amplification Attacks

In the previous section, we have shown that we can have an influence on the amplifier landscape. As such, we introduce next steps attackers may take once we "fix" all the protocols that have been documented to be vulnerable for amplification attacks [33]. Given the connection-less nature of UDP, it comes as no real surprise that UDP-based protocols may allow for amplification attacks.

In this section, we analyze to what extent TCP allows for amplification attacks similar to the UDP-based attacks. TCP is a connection-oriented protocol, in which early on (i.e., during the handshake) the IP addresses of both communication parties are implicitly verified via

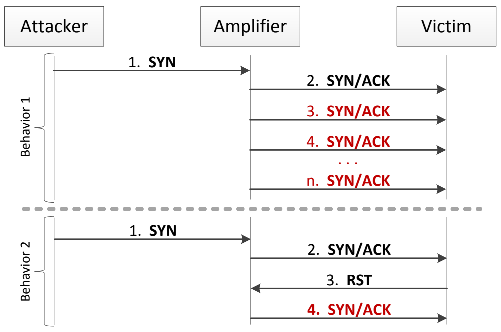

Figure 4: Amplification abuse in the TCP handshake

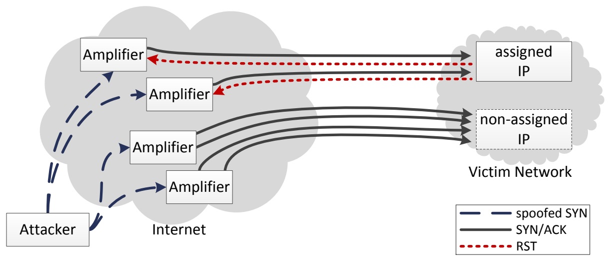

Figure 5: Attacker's choice of targets

initially-random TCP sequence numbers. We evaluate how TCP can be abused for amplification regardless of the TCP three-way-handshake.

From the viewpoint of the attackers, abusing TCP brings multiple benefits. First, providers cannot easily block or filter TCP traffic related to well-known protocols (such as HTTP), as compared to protocols that are less critical (such as CharGen, QOTD, or SSDP). In addition, it is hard to distinguish attacks from normal traffic in a stream of TCP control segments, while providers can deploy payload-based filters for attack traffic from many UDP-based protocols. Lastly, there are millions of potential TCP amplifiers out there and "fixing" them seems like an infeasible operation.

4.1 TCP Amplification Background

TCP initiates a connection with a three-way-handshake, which works as follows: a client willing to start a TCP connection sends a `SYN` segment to a server and this packet contains a random sequence number seq_A. If the server is willing to accept the client, it responds with a `SYN/ACK` segment, in which the acknowledgment number is set to $seq_A + 1$ and a random sequence number seq_B is added as well. In the third step, the client completes the connection setup by sending a final `ACK` to the server where the sequence number is set to $seq_A + 1$ and the acknowledgement number is set to $seq_B + 1$.

At first sight, TCP thus does not allow amplification: all segments are of the same size and no data bytes are exchanged before the handshake is finished. Assuming that the server draws TCP sequence numbers at random, there is no practical way to complete the handshake with IP-spoofed traffic. If the client address is spoofed, theoretically only one single `SYN/ACK` is sent to a potential victim. While this allows to reflect traffic, it does not amplify the traffic and therefore does not attack a victim with more bytes than sent by an attacker.

In practice, though, TCP connections encounter packet loss. TCP stacks thus deploy segment retransmissions, i.e., they retransmit segments that have not been acknowledged by the other party. We noticed that retransmissions also occur *during the handshake*. Many popular TCP stacks resend `SYN/ACK` segments until: (i) an `ACK` is received (and the connection is successfully established), (ii) a threshold is met (and the connection times out), or (iii) the connection is closed by the client (e.g., via a `RST` segment). In face of amplification attacks, this is problematic, as the client's IP address is not validated until the handshake is complete.

Figure 4 illustrates two typical behaviors of the TCP handshake. The first behavior illustrates a server repeatedly sending `SYN/ACK` segments, resulting in a TCP amplification. The second behavior points out a way for amplification even though the client (i.e., the victim) sends a `RST` segment to tear down the (not-yet-existing) TCP connection. In principle, this instructs servers (i.e, amplifiers) to stop sending any further `SYN/ACK` segments. We measured if hosts obey to this behavior.

4.2 Measuring TCP Amplification

As a first step to estimate the scope of this problem, we measure how the TCP stacks implement retransmissions during the TCP handshake. We perform an Internet-wide `SYN` scan and record the replies for further analysis. Our scanner does not complete the handshakes (i.e., we do not send `ACK` segments). With this, we aim to mimic the behavior of a system under an amplification attack that did not initiate the TCP connection in question. If TCP segments arrive that do not belong to any (half-)open connection (such as the reflected `SYN/ACK` segments in our scenario), TCP stacks either i) ignore these segments, or ii) respond with a `RST` segment, asking the other side to abort the TCP connection.

A victim, however, might not able to respond with `RST` packets, e.g., when it is already suffering from overload. Similarly, an attacker does not necessarily need to steer her attack against assigned IP addresses as shown in Figure 5. That is, the attacker can target an unassigned IP address so that there is no host that responds with `RST` segments. As a result, the capacity of the victim's net-

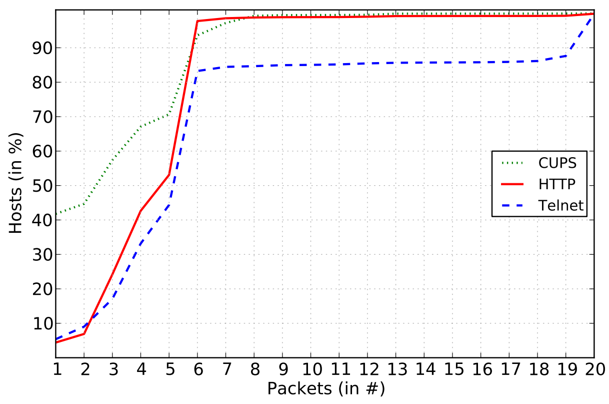

Figure 6: HTTP, Telnet, and CUPS without RST

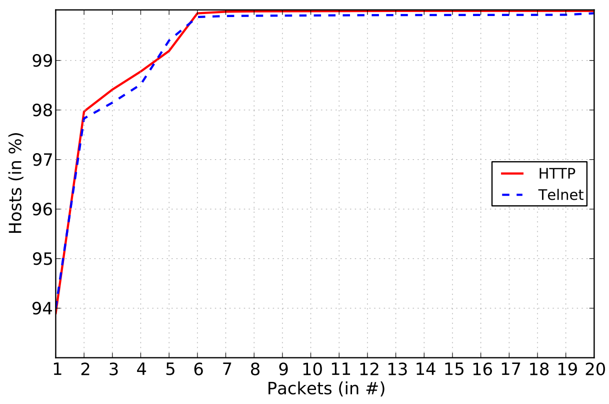

Figure 7: HTTP and Telnet with RST

work is overloaded by SYN/ACK segments. Addressing arbitrary IP addresses in the target network thus allows an attacker to control that no RST responses will be sent, leading to higher impact of an amplification attack.

TCP Scanning. We run two separated TCP scans to mimic both behaviors. In the first scan, we only send SYN segments and do not send anything back when receiving SYN/ACK segments. In the second scan, we "acknowledge" each incoming SYN/ACK segment with a RST segment. We performed the first scan for two popular protocols (i.e., HTTP and Telnet) and one printer-oriented protocol (CUPS). We chose these protocols, as according to Internet Census 2012, HTTP and Telnet yield a high number of reachable hosts [40]. In addition, to evaluate the TCP behavior of printers, we chose to scan CUPS.

In total, 66,785,451 HTTP hosts, 23,519,493 Telnet hosts, and 1,845,346 CUPS hosts replied to our requests. Figure 6 shows the results of the first scans as a CDF. The graph outlines that 6% of all HTTP hosts reply with a single SYN/ACK response and 24% of the hosts send at most three packets. A rise can be observed in-between 3 and 4 packets, meaning that 42% of the hosts reply to our SYN requests with 4 packets or less. We find the highest rise for 5 to 6 packets as 53.1% of all hosts send at most five SYN/ACK segments, but already 97.8% send six segments (or less). That is, 46.9% of the reachable HTTP hosts allow an amplification factor of 6 or higher.

Similar trends can be observed for the Telnet protocol. We find 55.6% of all Telnet hosts to enable an amplification attack with factor 6 or higher. In contrast to HTTP, about 12.2% of the Telnet hosts (more than 2.8 million systems) amplify requests even by factor 20. CUPS hosts, on the contrary, show less severe amplification rates. About 41.8% of the CUPS hosts replied with only a single packet and, in general, the number of segments sent by CUPS hosts is lower.

Figure 7 shows the results of our second scan (with RST). Indeed, sending RST tremendously changes the behavior of TCP stacks. In contrast to our first scan only 0.03% of Telnet hosts allow amplification by factor 20 which is a negligible number of 7,247 hosts. Almost 94% of all Telnet hosts send only one SYN/ACK packet in response; the same number applies for HTTP. A minor fraction of hosts keep resending SYN/ACK segments. We count 506,476 devices for HTTP and 114,157 devices for Telnet that send six SYN/ACK segments. Still, for the attack victim, replying with RST segments significantly reduces the impact for TCP amplification attacks and thus constitutes a potential proactive countermeasure.

4.3 Categorizing TCP Amplifiers

In this subsection, we aim to categorize the most prevalent behaviors that we have identified in the previous sections. Figures 6 and 7 have revealed significant groups of hosts that reply with the same number of TCP segments. Table 6 summarizes how many hosts belong to each of these groups. What kinds of systems are vulnerable to amplification attacks? To answer this question, we fingerprint the selected groups by re-using our TCP-based scans to obtain information via FTP, HTTP, HTTPS, SSH, and Telnet.

First, we classify the HTTP hosts that sent exactly six segments. Of the systems for which we obtained a fingerprint, about 88% (in total 3,228,000 hosts) run a Linux or Unix OS. Manual inspection has revealed that many of them are routers or embedded devices often running an FTP server (e.g., for a NAS). Other devices were vendor-specific, such as the ZynOS operating system. We also found numerous MikroTik devices and a smaller group of TP-LINK routers and D-Link devices. In contrast, there are only 0.2% Windows amplifiers in this group of hosts, hence this group of TCP amplifiers mainly consists of routers and various kinds of embedded devices.

Table 6: Hosts vulnerable to TCP amplification

	Number of response segments			
Scan	3	4	6	20
Without RST:				
CUPS	12.7%	9.7%	22.9%	0.0005%
	234,478	179,069	422,622	9
HTTP	17.3%	18.5%	44.7%	0.6%
	11,558,250	12,322,327	29,834,824	395,361
Telnet	8.1%	16.1%	38.9%	12.2%
	1,899,095	3,780,499	9,147,151	2,872,878
With RST:				
HTTP	0.44%	0.36%	0.76%	0.0005%
	294,535	243,028	506,476	349
Telnet	0.32%	0.37%	0.47%	0.03%
	76,774	88,504	114,157	7,247

In auxiliary tests, we have measured that Windows hosts (Windows 7, Vista, and XP) send four or five `SYN/ACK` segments in response—depending on the WINSOCK implementation. Although this amplification is not negligible, it is significantly lower than for other devices.

Second, we determine what kind of Telnet hosts sent six packets. Again, Unix and Linux are predominant as about 115 times more hosts in this group run Unix or Linux compared to the number of devices running Windows. Many of these hosts (49%) are routers, while other occurring devices are media servers, network cameras, digital video recorders, or VoIP phones.

Third, we analyze the Telnet hosts that sent 20 `SYN/ACK` segments. We found that 84.3% of all fingerprintable hosts in this group are routers or embedded devices. These embedded hosts—often based on MIPS or ARM architecture—include devices such as Raspberry Pis and printers. We found 86.1% of the devices to utilize the embedded web server *Allegro RomPager* and 37.5% to be manufactured by TP-LINK. In the remaining hosts, we also identified networking devices running ZynOS, ClearOS, or Cisco IOS. Typical desktop computers are negligible in this group: Windows is installed on 0.3% and MacOS on 0.0005% of these systems.

Lastly, we investigate the Telnet hosts that sent more than 20 `SYN/ACK` segments (21,981 hosts with an average of 971 response segments). Most of these hosts (87.9%) were found to be business and consumer routing devices of which 34% were running the embedded web server *GoAhead-Webs*. We find 50.4% of these devices to be a specific ATM Integrated Access device manufactured by RAD. Another 13.2% of the devices utilized the web server *Allegro RomPager* that we find to be associated to devices of manufacturers such as TP-LINK and ZyXEL. More information can be found in a further paper [20].

We conclude that amplification factors of 20 and more are largely caused by embedded devices and routers. We have contacted the vendors and wait for their feedback regarding these vulnerabilities.

5 Spoofer Identification

IP address spoofing is the root cause for amplification attacks, as it enables attackers to specify arbitrary targets that are flooded with reflected traffic. The Internet community addressed this issue as early as in May 2000 and suggested that—whenever possible—spoofed traffic should be blocked at the network edge [13]. However, as the attacks in practice have shown, spoofing still seems to be possible—yet it is unclear to what extent.

Up to now, the most powerful resource for tracking networks that allow spoofing is the Spoofer Project [36]. The project offers a client software that one can use to test if the own network filters IP-spoofed packets. Yet, such measurements require volunteers who download, compile/install, and run a client software. Aggregating user measurements in a study in July 2013, Beverly et al. show that about 610 of the 2582 tested ASes allow IP spoofing (at least partially, i.e., in some of their announced IP prefixes) [4]. On relative scale, however, less than 5% of the total number of ASes were tested. In other words, for more than 95% of the ASes it remains unclear if they support IP spoofing.

5.1 Remote Spoofer Test

Ideally, one would have a methodology to track networks that allow IP spoofing without the need for individuals running manual or tool-based tests from within the network. Such *remote tests* would boost the measurement coverage, so that we can alert administrators about potential misconfigurations that permit IP spoofing in their networks. We deploy such a large-scale experiment that enables us to identify thousands of ASes that support IP spoofing—from remote. Our DNS-based technique was first mentioned by Mauch on the NANOG mailing list in August 2013 [16]. It relies on public DNS proxies (or DNS stub resolvers—we will refer to "proxy" in the following) that have a broken networking implementation.

Figure 8 describes the core idea of the technique. The party that wants to identify spoofers (i.e., us) controls an Internet-scale scanner S and a name server that is authoritative for a domain suffix d_{suf}. In our case the domain d_{suf} is `scan.syssec.rub.de`. Note that we do not have control over devices on the right hand side, i.e., the DNS resolver and the optional DNS proxy, respectively. In step (1), the scanner S sends a DNS `A` lookup for domain d to an open resolver P. The domain d uses d_{suf} as domain suffix, but is specifically crafted for each scanning target. That is, S encodes a hex-formatted IP address of the scanning target P in the domain. This allows us to tell from the DNS response to which IP address we have sent the corresponding DNS request. In addition, to avoid caching

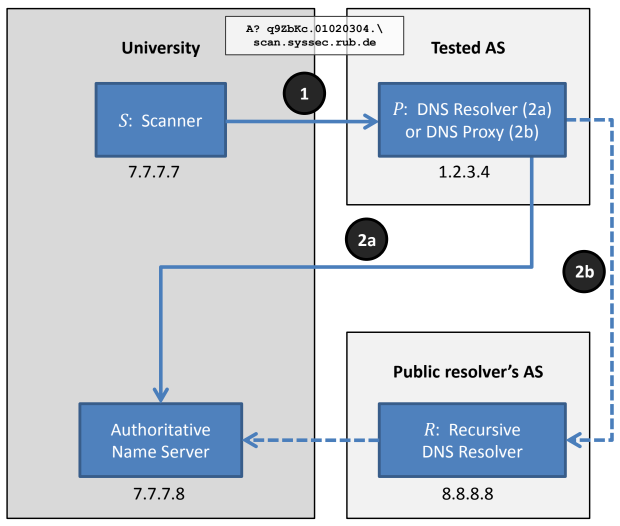

Figure 8: Setup to scan for spoofing-enabled networks. *R* is optional if *P* is an iterative resolver (step 2a) and is only used if *P* is a DNS proxy (step 2b)

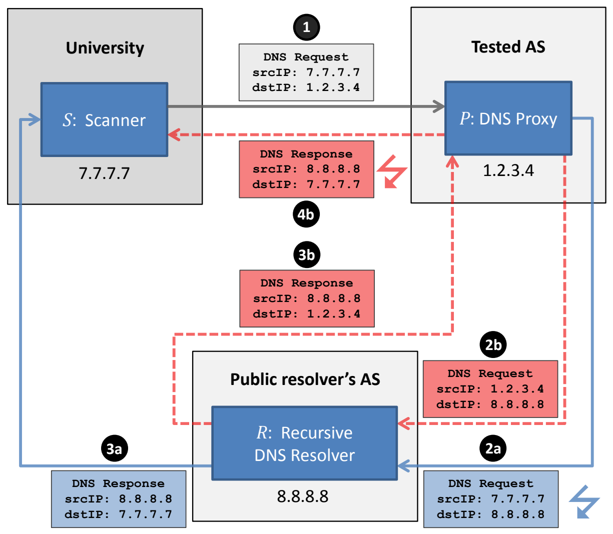

Figure 9: Network overview illustrating the two possible paths for DNS requests and responses when receiving responses for which the replying IP address did not match the IP address that was encoded in the requested domain

effects, we encode a random value in *d* that changes per request. In our case, *d* for target IP address 1.2.3.4 looks like `q9ZbKc.01020304.scan.syssec.rub.de`, whereas “q9ZbKc” is the random domain prefix and “01020304” the hex-formatted IP address.

When sending the request to *P*, one has to keep in mind that *P* may have different roles. If the scanned target *P* is a public recursive DNS resolver, *P* iteratively resolves the domain name by contacting the authoritative name servers down the domain tree as summarized in step (2a). For the purposes of our experiments such recursive resolvers are not important because they do not forward requests or responses. As we will show later, though, our technique to identify spoofing-enabled networks is based on the assumption that systems forward requests or responses. Quite often, *P* is not a resolver but a DNS proxy that forwards the DNS communication from a client (i.e., our scanner) to an iterative resolver *R*, as illustrated in step (2b).

We now leverage the fact that some DNS proxies do not correctly change the IP addresses when forwarding the request. In principle, to forward the DNS lookup to the resolver, the proxy *P* needs to change both the source and destination IP address of the request: it switches the source from *S* to its own address and the destination from its own address to *R*. Similarly, to forward the DNS response to the client, *P* changes the source from *R* to *P* and the destination from *P* to *S*. However, we encountered DNS proxies that do not change the addresses correctly. That is, we received DNS responses for which the replying IP address did not match the IP address that was encoded in the requested domain *d*. Instead, when *S* receives the response, the source IP address is set to *R*.

There are a few potential explanations for this observation. One is that *P* is a multi-homed system, i.e., has multiple interfaces with IP addresses in different networks. In many cases, though, proxies were using well-known resolvers (such as Google DNS) or resolvers in different ASes, excluding this possibility. Another explanation is that these devices have broken networking implementations, which cause the packets to have “wrong” IP header information when being forwarded. This could, e.g., be caused by broken Network Address Translation (NAT) implementations or faulty DNS proxy software.

Figure 9 illustrates the corresponding network situation when we receive a DNS response for which the source IP address of the UDP packet does not match the IP address that we encoded in the domain name of the DNS request. When sending a DNS request to the proxy (step 1), either *P* does not change the source address when forwarding the request to resolver *R* as outlined in step (2a) (i.e., the proxy effectively impersonates the sender *S*) so that *R* directly responds to *S* (step 3a). Alternatively, the proxy forwards the request to the resolver *R* (step 2b), obtains a DNS response (step 3b), and does not change the source address when forwarding the response to the sender *S* (step 4b), thus impersonates the resolver *R*. Either way, if *R* and *S* are not within *P*'s AS, then the proxy *P* is located in a network that permits the transmission of spoofed IP addresses. Both behaviors cause typical DNS clients to fail the resolution, as the DNS response comes from an unexpected IP address.

Table 7: Number of misconfigured DNS resolvers P and the corresponding Autonomous Systems

Filter	$\#P$	$\#AS_P$	Top 3 Countries
Top 4 Resolver	42,691	301	BR (52%), IT (11%), HU (10%)
Top 10 Resolver	45,072	352	BR (53%), IT (10%), HU (9%)
Distinct AS	170,451	2,692	CN (55%), BR (17%), RU (5%)
Distinct AS / # $\mathrm{resp}_R > 1$	161,988	2,063	CN (55%), BR (18%), RU (5%)
Distinct AS / # $\mathrm{resp}_R \geq 10$	137,075	870	CN (53%), BR (20%), RU (6%)

Clients will not even receive the replies if their network is protected by a stateful firewall, which drop UDP packets unrelated to any UDP stream known to the firewall. Unfortunately, we could not examine in detail which part of the forwarding was broken, as we did not control any of the recursive resolvers that the spoofing proxies used.

5.2 Finding Spoofing-Enabled Networks

While performing our Internet-wide scans, we observed a mismatch of source IP address and encoded target address for more than 2.2% of all responsive DNS servers, resulting in a total of 581,777 DNS proxies which redirect incoming requests to 225,888 distinct recursive DNS resolvers. To explore these misbehaving DNS proxies and the corresponding ASes in more detail, we enumerate the number of ASes permitting IP address spoofing using the following filtering methods:

(i) Our most conservative estimation is based only on responses from four commonly-used open resolvers operated by Google (i.e., 8.8.8.8 and 8.8.4.4) and OpenDNS (208.67.222.222 and 208.67.220.220). These servers ("Top 4") are a subset of the servers in the second approach.

(ii) Less conservative, we take into account DNS responses of the most popular ten resolvers ranked by the number of proxies using them ("Top 10").

(iii) Lastly, we focus on proxies for which $AS(S) \neq AS(P) \neq AS(R)$ applies. In other words, the proxy is not located in the same AS as both the sender S and the resolver R, and thus is spoofing the identity of one of these identities.

Table 7 illustrates the results obtained for each filtering method. In total, 7.7% of the potentially-spoofing DNS proxies forward the DNS requests to the Top 10 well-known resolvers (filter (ii)), resulting in 352 distinct ASes the proxies are located in. When limiting our focus to the Top 4 resolvers (filter (i)), we still identify 301 different ASes that permit spoofed traffic. Furthermore, we find 29.3% of all proxies to be located in different ASes than the sender S and the resolvers R (filter (iii)), resulting in 2,692 ASes permitting the proxies to either spoof the IP address of S or R.

Of the 225,888 individual resolvers R we find 50.7% utilized by multiple DNS proxies. To exclude potentially multi-homed systems with multiple interfaces in distinct ASes, we restrict the set of resolvers to those which responded to requests from multiple proxies and find 2,063 ASes that allow spoofed traffic. When further filtering the set to resolvers that replied to at least ten different proxies, we still identify 870 ASes permitting spoofing. Using our remote test, we can thus identify more spoofing-enabled ASes than the current state-of-the-art manual analyses performed by the Spoofer Project [36].

5.3 Fingerprinting IP-Spoofing Devices

Lastly, we want to understand what type of devices follow the weird practice of spoofing IP addresses while forwarding DNS requests/responses. For this, we use our TCP-based fingerprints to classify the 42,691 devices that used Google DNS or OpenDNS as iterative resolver. Of these devices, 6,120 devices replied to our TCP requests and 5,674 resolvers provided information suitable for fingerprinting. In total, we find 3,033 devices running the Dropbear SSH daemon, particularly employed on embedded devices. We also identify 1,437 MikroTik routers to be forwarding requests specifically to the Google DNS servers. Further 540 devices of the manufacturer Airlive perform similar behavior.

We achieve similar results when fingerprinting the hosts of the other filtering methods (see previous subsection). We again find Dropbear, MikroTik, and Airlive to appear frequently. We assume that these devices have either bad NAT rules or erroneous DNS proxy implementations. However, requests for more specific information from the vendors remained unanswered until now.

5.4 Remote Test Limitations

Our results show that DNS-based spoofing tests are a powerful resource to identify spoofing-enabled networks. One inherent limitation of this approach is, though, that such tests do only reveal the fact *that* (and not *if*) a network allows IP spoofing. We leave it up to future work to test if the tests can be expanded accordingly. For example, we could scan for DNS proxies that can be fingerprinted as systems that typically spoof IP addresses. In addition, collaborating with the recursive resolvers (such as OpenDNS or Google DNS) may reveal further insights about the spoofing systems. Lastly, given the large number of hosts running other protocols than DNS, it may be possible to use further protocols for similar remote spoofing tests.

6 Related Work

Our work was inspired by the analysis of amplification attacks by Rossow [33]. He identified 14 UDP-based network protocols that are vulnerable to amplification attacks and gave a thorough overview of countermeasures. We continue this line of research and classify the amplifiers. We show in an Internet-wide NTP amplifier notification initiative that the threats can be mitigated by cooperation within the security community. We furthermore investigate to what extent TCP-based amplification attacks are possible. Lastly, we provide an overview of spoofing-enabled networks. Our work is thus a thorough and novel extension of Rossow's initial analysis.

We group further related works by their topic:

TCP Amplification. To the best of our knowledge, we are the first to evaluate the amplification potential of the TCP three-way handshake. Prior work on TCP amplification has addressed guessable TCP sequence numbers, which in principle allow to establish TCP connections with spoofed packets [2, 31]. In addition, Paxson et al. looked at amplification in Transactional TCP (T/TCP)—which has very low popularity though [31]. Lastly, well-known stateful TCP attacks like the FTP bounce attack also allow for amplification [6]. Many of these attacks have been largely fixed with secure TCP stack implementations or by hardening certain protocols (e.g., FTP). The amplification vulnerabilities that we discovered in the TCP three-way handshake may again require improvements to TCP stacks.

Internet-Wide Scanning. Durumeric et al. presented ZMap, a publicly-available tool optimized for Internet-wide scans [11]. In fact, we leverage most of their proposed techniques and implemented their guidelines also for our custom scanner. Zhang et al. used Internet-wide scans to correlate the mismanagement and the maliciousness of networks [44]. They find networks that host open recursive DNS resolvers highly correlate to other malicious activities (such as spamming) initiated from these networks. Our work is orthogonal, as we follow a proactive approach to cooperate with the providers in order to get the vulnerabilities fixed. Two non-academic projects deployed by Mauch, the OpenNTPProject [30] and Open Resolver Project [29], also address the problems of amplification sources from a practical point of view. We have collaborated with Mauch to inform administrators of NTP servers vulnerable to the `monlist` amplification and are grateful for his support.

DDoS Attack Types. An alternative way to launch powerful DDoS attacks are networks of remotely-controllable bots. Büscher and Holz analyze DirtJumper, a botnet family with the specific task to perform DDoS attacks by abusing the Internet connection of infected desktop computers [5]. 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 [19]. 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 [38]. All these attacks rely on bots, while our threat model only assumes that an attacker has any spoofing-enabled Internet uplink. Although the amplification DDoS attacks primarily try to congest bandwidth of a single victim, they can possibly be combined with the aforementioned techniques.

7 Conclusion

We have confirmed that amplification attacks remain a major Internet security issue—not only for UDP-based protocols. We identified TCP as an alternative source for amplification—despite its three-way-handshake protocol. We find millions of systems with TCP stacks that can be abused to amplify TCP traffic by a factor of 20x or higher. Our work revealed a tremendous number of potential amplification sources for both UDP and TCP-based protocols and classified these systems. During a first-ever large-scale notification campaign, we have observed a significant decrease in the number of amplifiers for NTP, giving hope for future attempts in fixing protocols that have similarly-severe amplification vulnerabilities. Finally, our remote spoofing test has identified more than 2,000 networks that do not use proper egress filtering—indicating that it is still a long way to go until we will have a spoofing-free Internet.

Acknowledgment

We would like to thank the anonymous reviewers and Jared Mauch for their constructive and valuable comments. This work was supported by the German Federal Ministry of Education and Research (BMBF Grants 16BY1110 (MoBE) and 16BY1201D (iAID)).

References

[1] BAILEY, M., COOKE, E., JAHANIAN, F., AND NAZARIO, J. The Internet Motion Sensor - A Distributed Blackhole Monitoring System. In *Symposium on Network and Distributed System Security (NDSS)* (2005).

[2] BELLOVIN, S. RFC 1984: Defending Against Sequence Number Attacks, 1996.

[3] BEVERLY, R., BERGER, A., HYUN, Y., AND CLAFFY, K. Understanding the Efficacy of Deployed Internet Source Address Validation Filtering. In *Internet Measurement Conference (IMC)* (2009).

[4] BEVERLY, R., KOGA, R., AND CLAFFY, K. Initial Longitudinal Analysis of IP Source Spoofing Capability on the Internet. *Internet Society* (2013).

[5] BÜSCHER, A., AND HOLZ, T. Tracking DDoS Attacks: Insights into the Business of Disrupting the Web. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2012).

[6] CENTER, C. C. FTP Bounce: Advisory CA-1997-27. `http://www.cert.org/advisories/CA-1997-27.html`, 1997.

[7] CISCO PSIRT. Cisco Network Time Protocol Distributed Reflective Denial of Service Vulnerability. `http://tools.cisco.com/security/center/content/CiscoSecurityNotice/CVE-2013-5211`, 2014.

[8] COMPUTER EMERGENCY RESPONSE TEAM. CERT advisory CA-1996-21: TCP SYN Flooding and IP Spoofing Attacks. `https://www.cert.org/historical/advisories/CA-1996-21.cfm`, 1996.

[9] DITTRICH, D. Distributed Denial of Service (DDoS) Attacks/tools. `http://staff.washington.edu/dittrich/misc/ddos/`, 2000.

[10] DURUMERIC, Z., BAILEY, M., AND HALDERMAN, J. A. An Internet-wide View of Internet-wide Scanning. In *USENIX Security Symposium* (2014).

[11] DURUMERIC, Z., WUSTROW, E., AND HALDERMAN, J. A. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *USENIX Security Symposium* (2013).

[12] EHRENKRANZ, T., AND LI, J. On the State of IP Spoofing Defense. *ACM Trans. Internet Technol. 9*, 2 (May 2009).

[13] FERGUSON, P., AND SENIE, D. BCP 38: Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing, 1998.

[14] FREILING, F. C., HOLZ, T., AND WICHERSKI, G. Botnet Tracking: Exploring a Root-cause Methodology to Prevent Distributed Denial-of-service Attacks. In *European Symposium on Research in Computer Security (ESORICS)* (2005).

[15] IOANNIDIS, J., AND BELLOVIN, S. M. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Symposium on Network and Distributed System Security (NDSS)* (2002).

[16] J. MAUCH. `http://seclists.org/nanog/2013/Aug/132`, August 2013.

[17] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *World Wide Web Conference (WWW)* (2002).

[18] KANDULA, S., KATABI, D., JACOB, M., AND BERGER, A. Botz-4-sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In *USENIX Symposium on Networked Systems Design and Implementation* (2005).

[19] KANG, M. S., LEE, S. B., AND GLIGOR, V. D. The Crossfire Attack. In *Proceedings of IEEE Security and Privacy (S&P)* (San Francisco, CA, USA, 2013).

[20] KÜHRER, M., HUPPERICH, T., ROSSOW, C., AND HOLZ, T. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks. In *USENIX Workshop on Offensive Technologies (WOOT)* (2014).

[21] M. PRINCE; CLOUDFLARE, INC. `http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet`, March 2013.

[22] M. PRINCE; CLOUDFLARE, INC. `http://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack`, February 2014.

[23] MAXMIND GEOIP DATABASE. `http://www.maxmind.com/en/ip-location`, 2014.

[24] MIRKOVIC, J., DIETRICH, S., DITTRICH, D., AND REIHER, P. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, 2004.

[25] MITRE. CVE-2013-5211. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-5211`, 2013.

[26] MOORE, D., VOELKER, G. M., AND SAVAGE, S. Inferring Internet Denial-of-service Activity. In *USENIX Security Symposium* (2001).

[27] NETWORK MAPPER. `http://nmap.org/`, 2014.

[28] NTP POOL PROJECT. `http://www.pool.ntp.org`, 2014.

[29] OPEN RESOLVER PROJECT. `http://OpenResolverProject.org/`, 2013.

[30] OPENNTP SCANNING PROJECT. `http://OpenNTPProject.org/`, 2014.

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

[32] PORRAS, PHILLIP AND SAIDI, HASSEN AND YEGNESWARAN, VINOD. Technical Report: Conficker C Analysis. `http://mtc.sri.com/Conficker/`, 2009.

[33] ROSSOW, C. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Symposium on Network and Distributed System Security (NDSS)* (2014).

[34] SCHUBA, C. L., KRSUL, I. V., KUHN, M. G., SPAFFORD, E. H., SUNDARAM, A., AND ZAMBONI, D. Analysis of a Denial of Service Attack on TCP. In *IEEE S&P* (1997).

[35] SHADOWSERVER FOUNDATION. `http://shadowserver.org/`, 2014.

[36] SPOOFER PROJECT. `http://spoofer.cmand.org/`, 2014.

[37] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R., KRUEGEL, C., AND VIGNA, G. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS)* (2009).

[38] STUDER, A., AND PERRIG, A. The Coremelt Attack. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)* (Saint Malo, France, September 2009).

[39] SYMANTEC CORPORATION. Hackers Spend Christmas Break Launching Large Scale NTP-Reflection Attacks. `http://www.symantec.com/connect/blogs/hackers-spend-christmas-break-launching-large-scale-ntp-reflection-attacks`, 2013.

[40] THE INTERNET CENSUS 2012. Port scanning /0 using insecure embedded devices. `http://internetcensus2012.bitbucket.org/paper.html`, 2012.

[41] TRUSTED INTRODUCER. `http://trusted-introducer.org/`, 2014.

[42] US-CERT. NTP Amplification Attacks Using CVE-2013-5211. `http://www.us-cert.gov/ncas/alerts/TA14-013A`, 2014.

[43] YAAR, A., PERRIG, A., AND SONG, D. Pi: A Path Identification Mechanism to Defend Against DDoS Attacks. In *IEEE S&P* (2003).

[44] ZHANG, J., DURUMERIC, Z., BAILEY, M., LIU, M., AND KARIR, M. On the Mismanagement and Maliciousness of Networks. In *Symposium on Network and Distributed System Security (NDSS)* (2014).