Weaponizing Middleboxes for TCP Reflected Amplification

Kevin Bock* Abdulrahman Alaraj† Yair Fax* Kyle Hurley* Eric Wustrow† Dave Levin*
*University of Maryland †University of Colorado Boulder

Abstract

Reflective amplification attacks are a powerful tool in the arsenal of a DDoS attacker, but to date have almost exclusively targeted UDP-based protocols. In this paper, we demonstrate that non-trivial TCP-based amplification is possible and can be orders of magnitude more effective than well-known UDP-based amplification. By taking advantage of TCP-non-compliance in network middleboxes, we show that attackers can induce middleboxes to respond and amplify network traffic. With the novel application of a recent genetic algorithm, we discover and maximize the efficacy of new TCP-based reflective amplification attacks, and present several packet sequences that cause network middleboxes to respond with substantially more packets than we send.

We scanned the entire IPv4 Internet to measure how many IP addresses permit reflected amplification. We find hundreds of thousands of IP addresses that offer amplification factors greater than 100×. Through our Internet-wide measurements, we explore several open questions regarding DoS attacks, including the root cause of so-called "mega amplifiers". We also report on network phenomena that causes some of the TCP-based attacks to be so effective as to technically have *infinite* amplification factor (after the attacker sends a constant number of bytes, the reflector generates traffic indefinitely). We have made our code publicly available.

1 Introduction

Volume-based distributed denial of service (DDoS) attacks operate by producing more traffic at a victim's network than its capacity permits, resulting in decreased throughput and limited availability. An important component in the arsenal of a DDoS attacker is the ability to *amplify* its traffic. Instead of sending traffic directly to a victim V, the attacker spoofs V's source address, sends b bytes to some *amplifier* host A, who then "replies" to V with $\alpha \cdot b$ bytes for some $\alpha > 1$. In this manner, the attacker hides its IP address(es) from the victim, making it difficult to simply filter the attack traffic at a firewall, and increases its effective capacity by the *amplification factor* α.

Some reflected amplification attacks can elicit impressive amplification factors. Among the most notable, DNS has been shown to have an amplification factor of 54, while NTP offers up to 556.9 [32]. Misconfigured Memcached [37] servers can provide amplifications over 51,000 [8, 40], and were used against Github in 2018 in the largest known DDoS attack to date, achieving 1.35 Tbps at peak [14].

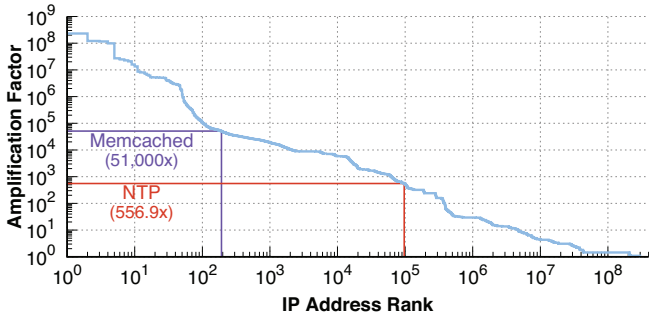


Figure 1: The maximum amplification factor we obtained per IPv4 address, based on several Internet-wide scans. (Note: the axes are log-scale.)

To date, almost all reflected amplification attacks have leveraged UDP. In fact, to the best of our knowledge, there are *no* known TCP-based reflected amplification attacks that send beyond a single `SYN` packet.[1] This is because such attacks appear virtually impossible: to go beyond the `SYN` would seem to require an attacker to (1) guess the amplifier's 32-bit initial sequence number (ISN) in their `SYN+ACK` packet[2] and (2) prevent the victim from responding to the amplifier with a `RST` [23].

In this paper, we show that it is indeed possible to launch reflected amplification attacks with TCP beyond a single `SYN` packet without having to guess initial sequence numbers. The key insight is to not elicit responses from the destination, but rather from *middleboxes* on the path to the destination.

Many middleboxes (especially nation-state censors) inject block pages or other content (such as `RST` packets) [13, 31, 42, 46] into established TCP connections when they detect forbidden requests. Moreover, because middleboxes cannot rely on seeing all packets in a connection [7], they are often designed to operate even when they see only one side of the connection. Our attacks tend to leverage *non-compliant middleboxes* that respond without having to observe both ISNs. Our measurements show that such middleboxes are surprisingly common on today's Internet, and that they can

[1] We discuss *non-reflected* TCP-based amplification attacks in Section 8.
[2] We will use + to denote when a single packet has multiple TCP flags set.

lead to amplification factors surpassing even many of the best UDP-based amplification factors to date.

We introduce a novel application of a recent network-based genetic algorithm [6] that *discovers* sequences of TCP packets that elicit large amplification factors from middleboxes.

We perform a series of IPv4-wide scans of the Internet using ZMap [10], to identify how many hosts can serve as amplifiers and quantify their amplification factor. Figure 1 provides an overview of the maximum amplification factor we were able to get from all IP addresses after several Internet-wide scans. We find 386,187 IP addresses that yield an amplification factor of at least 100×; 97,079 IP addresses that elicit a larger amplification factor than the infamous NTP attack [32], and over 192 IP addresses that responded with a higher amplification factor than Memcached [8].

Compared to `SYN`-only reflective amplification attacks, our attack identifies two orders of magnitude more IP addresses [15, 16], and we also find amplification factors above 2,500×.

In fact, we find many hosts that effectively have an *infinite* amplification: in response to one or two attack packets, these machines respond at their full capacity indefinitely (barring packet drops) without any additional attacker involvement. Czyz et al. [9] observed similar behavior when studying NTP amplification, and called such hosts "mega-amplifiers." We at last answer the open question of why some hosts provide such abnormally high amplification factors: we show that many are actually sustained *by the victims themselves*, and others are due to routing loops.

Collectively, our results show that there is significant, untapped potential for TCP-based reflective amplification attacks. To enable this new area of study, we have made our code publicly available at `https://geneva.cs.umd.edu/weaponizing`.

Contributions We make the following contributions:

- We introduce a novel application of genetic algorithms to discover and maximize the efficacy of TCP-based reflective amplification attacks, and identify 5 attacks in total.
- We scan the IPv4 Internet to determine how many IP addresses can be used as TCP-based amplifiers, and their amplification factor.
- We confirm that these amplified responses typically come from network middleboxes, including government censorship infrastructure and corporate firewalls.
- We resolve the open question of the root causes of "mega-amplifiers." We attribute them to infinite routing loops and what we call "victim-sustained amplification", in which victims' default responses (`RST`s) actually induce the reflector to send more data without additional effort from the attacker, leading to virtually infinite amplification.

The rest of this paper is organized as follows. We review background in §2. In §3, we present novel techniques for discovering new TCP-based amplification attacks, and the results from applying these techniques to live censoring middleboxes. Next, we describe our methodology (§4) and results (§5) from scanning the entire IPv4 Internet with our newfound attacks. We explore "mega-amplifiers" in §6. We discuss ethical considerations and our responsible disclosure in §7, related work in §8, potential countermeasures in §9, and conclude in §10.

2 Background

Here, we define our threat model and review details of TCP and in-network middleboxes that are relevant to our attacks.

Threat Model To maximize the applicability of our attacks, we make very few assumptions about the adversary's capabilities. In particular, we assume a completely *off-path* attacker: it cannot eavesdrop, intercept, drop, or alter any packets other than the ones destined to it. We also assume that the attacker has the ability to source-spoof its victim's IP address. This would not be possible if the attacker's network performs *egress filtering*—that is, if it verified that the packets leaving its network had IP addresses originating from within its network—but egress filtering is still not yet widely deployed in practice [4, 15, 39].

TCP Basics To ensure in-order delivery of bytes, both ends of a TCP connection assign 32-bit *sequence numbers* to the bytes they send. TCP connections begin with a *three-way handshake*, during which the end-hosts inform one another of their (random) initial sequence number (ISN). In a standard three-way handshake, the client sends a `SYN` packet containing its $\text{ISN}_{\text{client}}$, to which the server responds with a `SYN+ACK` that contains both its own $\text{ISN}_{\text{server}}$ and $\text{ISN}_{\text{client}} + 1$ to acknowledge the client's ISN. Finally, the client acknowledges $\text{ISN}_{\text{server}}$ by including it (plus one) in an `ACK` packet. Following this, a typical client sends a `PSH+ACK` packet containing its application-layer data (e.g., an HTTP GET request).

For a TCP connection to complete, the ISNs must be acknowledged with perfect accuracy. If the client were to send an `ACK` acknowledging anything but $\text{ISN}_{\text{server}} + 1$, the server would not accept the connection.

TCP-based Reflection Attacks In a *reflection* attack, an adversary sends to a destination r a packet that spoofs the source IP address to be that of victim v. As a result, r will believe v sent the packet, and will send its response to v. Reflection can be useful to hide the attacker's identity from the victim, and is commonly used when the reflector r is also an amplifier, sending more data to v than r received from the attacker.

Note that an adversary within our threat model cannot feasibly complete a three-way handshake in a reflection attack. The adversary would send the `SYN` while source-spoofing v, and thus the server's `SYN+ACK`—with $\text{ISN}_{\text{server}}$—would be sent to v, not the attacker. To complete the handshake, the

attacker would have to send a source-spoofed ACK, but would only have 2^{-32} chance of guessing the correct ISN_{server}. Moreover, even if the adversary were to guess ISN_{server}, the victim (if online) will respond to the server's spurious SYN+ACK with a RST, thereby tearing down the connection at the server.

Given these challenges, prior work assumed that TCP-based reflection attacks were limited to the initial handshake, in which the attacker sends a source-spoofed SYN and does not try to guess the appropriate ACK, let alone send an application-layer PSH+ACK [15,16]. Kührer et al. [16] showed that a single TCP SYN can result in a surprising amount of amplification. Compliant servers amplify a small amount because they retransmit SYN+ACKs a handful of times, until they timeout, receive the appropriate ACK, or receive a RST from the victim. Kührer et al. also found a few non-compliant machines on the Internet that respond to SYNs with many more packets, affording a greater amplification [15,16].

In this work, we discover that *middleboxes* enable more sophisticated TCP-based reflected attacks beyond a single SYN. Compared to prior work, these new middlebox-enabled attacks yield even higher amplification rates and provide larger numbers of amplifiers that attackers can use.

Middleboxes A *middlebox* is an in-network device that sits on the path between two communicating end-hosts, and can monitor, filter, or transform packet streams in-flight. Unlike traditional network devices like routers and switches, middleboxes operate not only on packets' headers, but also on their payloads using Deep Packet Inspection (DPI).

Middleboxes have been used for myriad network functionality applications [2, 35, 44], including firewalls. Firewalls allow administrators to limit what content is viewable by end-hosts within their networks.

Some of the most widespread and pernicious deployments of firewall middleboxes are by *nation-state censors*, often in an attempt to suppress access to information. Censoring middleboxes are typically located at the nation's borders (or within the nation's ISPs), and are commonly deployed at massive scales so that they may monitor all traffic traversing the censoring nation-state [3,24,45].

Censoring firewalls typically identify forbidden keywords or domains in plaintext traffic, DNS requests, or TLS server name indication (SNI) fields. Once a censoring middlebox determines a connection should be censored, it can do so in different ways: by dropping offending packets [5], injecting RST packets to tear down the connection [6,42], injecting false DNS responses [42,46] or—critical to this work—by injecting *block pages* in response to forbidden HTTP requests [22,41].

Middleboxes often track the content of connections across multiple packets to handle re-ordered or dropped packets. However, middleboxes may not see packets in both directions. This is because the Internet can exhibit *route asymmetry*, whereby packets between two end-hosts may traverse different paths [26]. Consequently, a middlebox may only see one side of a TCP connection (e.g., the packets from client to server). To handle this asymmetry, middleboxes often implement non-compliant or partial TCP reassembly, allowing them to still block connections even though they don't see all of the packets in a connection.

Middleboxes' resilience to missing packets presents an opportunity to attackers: a reflecting attacker may not need to complete the three-way handshake so long as it can convince the middlebox that the handshake had been completed. Combined with the packets they inject—especially block pages—middleboxes could be attractive targets for reflected amplification. In the remainder of this paper, we show packet sequences that trick middleboxes into responding, and we show that middleboxes can yield very large amplification factors.

3 Discovering TCP-based Reflection Attacks

In this section, we present the first non-trivial, TCP-based reflected amplification attacks. We present a novel way to automatically discover new amplification attacks (§3.1), train it against a set of censoring middleboxes (§3.2), and report on the amplification attacks we discovered (§3.3).

3.1 Automated Discovery of Amplification

Our goal is to identify sequences of packets that will elicit amplified responses from middleboxes, without requiring us to establish a legitimate TCP connection or guess ISNs. This requires identifying non-compliant TCP behavior. Unlike UDP [9] or TCP SYN-based [16] reflected amplification attacks—which take advantage of weaknesses in protocol designs—we must find weaknesses in TCP *implementations*.

Recent efforts have created automated ways to identify input sequences that cause incorrect middlebox behavior [6,43]. In 2019, Bock et al. developed Geneva, an open-source automated tool for discovering packet manipulation sequences (called "strategies") to evade censorship [6]. Geneva uses a genetic algorithm to evolve censorship-evasion techniques by composing five packet-level actions: duplicate, tamper, fragment, drop, and send. Over a series of discrete "generations," Geneva tests dozens of packet manipulation strategies directly against real-world censors. Geneva evaluates strategies with a *fitness function*: a numeric score that captures how successful a given strategy is at evading censorship. Strategies that receive a higher score are more likely to survive and pass their "genetic code" to the next generation.

We make two modest changes to Geneva to find new amplification attacks against middleboxes:

Initial Packet Sequence Geneva operates by manipulating an existing packet sequence, such as a real client's packets as it browses the web. To discover new amplification attacks, we use a single PSH+ACK packet with a well-formed HTTP GET request with the Host: header set to a given URL (we describe which URLs we use in §3.2). We chose HTTP as

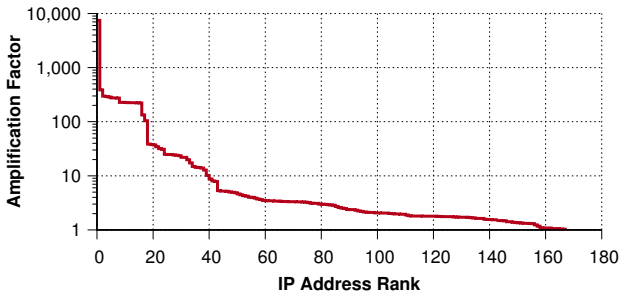


Figure 2: Rank order plot of maximum amplification factor from Quack-identified IP addresses. The maximum amplification factor was 7,455×.

the input traffic because recent work demonstrated both how widely deployed HTTP filtering middleboxes are [31] and that many HTTP censors inject large block pages in response to small web requests [41].

Fitness Function Our goal is to find packet sequences that maximize amplification from middleboxes. The straightforward approach would be to set the fitness function to the amplification factor itself (number of bytes received divided by the number of bytes sent). However, we found that this sometimes encourages Geneva to try to elicit many small (e.g., `SYN+ACK`) packets from the end-host, rather than larger (e.g., block page) packets from middleboxes. To encourage Geneva to elicit responses specifically from middleboxes, our fitness function is the amplification factor, but ignoring all incoming packets that have no application-level payload. This optimization applies only to the fitness function; we report on *all* bytes sent and received in our results.

3.2 Training Methodology

Geneva trains on live networks, and thus requires destination IP addresses to train against. To identify destination IP addresses that are likely to have middleboxes on the path from our measurement machine to them, we use data from Quack [41], a part of the Censored Planet [30] platform that performs active measurements of censorship. Quack regularly sends HTTP GET requests with potentially forbidden URLs in the `Host:` header to echo servers around the world, and detects injected censorship responses from middleboxes.

We use Quack's daily reports [27] to find endpoints that are likely to have middleboxes on the path, and the URLs likely to trigger them. We downloaded Quack's March 28th, 2020 dataset and extracted the IP addresses that experienced HTTP injection interference. This identified 209 IP addresses with active censoring middleboxes on their path, along with the offending URLs. We began training against them on March 29th.

To train Geneva with an IP address from Quack's data, we set the destination of the generated traffic to the IP address, and set the `Host:` header in the HTTP GET request to one of the URLs that triggered interference to this IP address.

We let Geneva train for 10 generations with an initial population of 1,000 randomly generated strategies[3]. Training took approximately 25 minutes per IP address. To limit our impact on the network, we spaced our experiments out over four days; we sent each end-host just 2.8 Kbps of traffic on average (comparable to Quack's scans).

Before each experiment, we repeated Quack's methodology to the destination IP address to confirm it is still experiencing interference, and we skipped IP addresses that we did not experience interference. During our experiments, 25 of the 209 IP addresses (11.9%) stopped responding or no longer experienced interference, consistent with the churn rates seen in Quack's original experiments [41]. This left 184 IP addresses with active censoring middleboxes that Geneva trained against. Next, we present the packet sequences Geneva discovered.

Strategy	Response %	Max Amplification
⟨`SYN`; `PSH+ACK`⟩	69.5%	7,455×
⟨`SYN`; `PSH`⟩	65.7%	24×
`PSH`	44.6%	14×
`PSH+ACK`	33.1%	21×
`SYN` (with GET)	11.4%	572×

Table 1: TCP-based reflected amplification attacks discovered against 184 Quack servers. Each packet with the `PSH` flag set includes an offending HTTP GET request in the payload.

3.3 Discovered Amplification Attacks

For 178 (96.7%) of the 184 IP addresses from the Quack dataset, Geneva found at least one packet sequence that elicited a response, and achieved an amplification factor greater than 1 for 169/178 (94.9%). Figure 2 shows the maximum amplification factors we discovered across all of these 169 hosts. Some of the middleboxes provided high amplification factors: 17 (9.5%) had greater than 100×, and the maximum amplification factor was 7,455×.

We identify five unique packet sequences that elicit responses and five additional modifications to improve amplification factor. We summarize them in Table 1 and describe them in turn below.

3.3.1 Amplifying Packet Sequences

⟨**`SYN; PSH+ACK`**⟩ The most successful strategy we discovered sends a `SYN` packet (with no payload) with sequence number s, followed by a second `PSH+ACK` packet containing sequence number $s+1$ and the forbidden GET request. Although this strategy comes at the cost of an entire additional packet, we find it to be highly effective at getting responses

[3] We forgo a full hyperparameter sweep to limit our impact on end hosts.

from middleboxes. It elicited responses from 128/184 (69.6%) of the middleboxes, with a maximum amplification factor of 7,455×.

From a middlebox's perspective, this packet sequence looks like a traditional TCP connection, missing the server's SYN+ACK and the client's ACK. As with normal TCP connections, the sequence number of the SYN is one less than the sequence number of the PSH+ACK. As discussed in §2, middleboxes must be resilient to asymmetric routes, so it is expected that they would respond while missing the server's SYN+ACK. We note this sequence omits the client's ACK in a typical handshake, though the PSH+ACK may suffice to replace it. Geneva tried adding the client's ACK, but eliminated it during training—in follow-up experiments, we verified that adding the ACK had no effect on how the middleboxes responded.

⟨**SYN; PSH**⟩ This sequence sends a SYN with sequence number s (and no payload) followed by a PSH with sequence number $s+1$ and the forbidden GET request as its payload. Note that this is the same as the ⟨SYN; PSH+ACK⟩ strategy, but with the ACK flag cleared in the second packet.

⟨SYN; PSH⟩ elicited responses from 121/184 (65.7%) of middleboxes, with a maximum amplification of 24×. Most (118, or 97.5%) of these also responded to the ⟨SYN; PSH+ACK⟩ sequence with the same amplification factors: those middleboxes appear not to be sensitive to the presence of the ACK flag on the packet containing the request. However, 10 middleboxes responded only when the ACK flag was set and 3 middleboxes responded only when it was not. We explore these differences more deeply with full IPv4 scans in §5.

We also explored if an additional ACK packet between the SYN packet and the PSH packet would improve response rate. Like with the ⟨SYN; PSH+ACK⟩ sequence, we found it had no effect on the middleboxes' responses.

PSH This sequence sends only a single packet: a PSH with the forbidden GET request. It elicited responses from 82 (44.6%) of middleboxes, with a maximum amplification factor of 14×. Note that this is the same as the ⟨SYN; PSH⟩ sequence, without the SYN. All but one (98.8%) of the middleboxes that responded to just the PSH also responded to ⟨SYN; PSH⟩, indicating that the SYN was not necessary. For those hosts, avoiding the SYN resulted in an increase in amplification factor.

PSH+ACK This also sends a single packet: a PSH+ACK with a forbidden GET request. No TCP-compliant host should respond to this packet with anything besides an empty RST, as there is no three-way handshake. Still, 61 (33.2%) middleboxes responded with injected responses, with a maximum amplification factor of 21×.

This strategy is identical to the ⟨SYN; PSH+ACK⟩ sequence, minus the SYN packet. We find that all of the middleboxes that responded to a lone PSH+ACK also responded to the ⟨SYN; PSH+ACK⟩, with the responses of the same size. For those hosts, sending the additional SYN strictly decreases the amplification factor.

Most (51, or 83.6%) of the middleboxes that responded to PSH+ACK also responded to PSH; these middleboxes' responses were the same for both strategies, indicating no change in amplification. 10 middleboxes responded to PSH+ACK but not to PSH; these gave PSH+ACK its greatest amplification factor. However, 31 middleboxes responded to PSH but not PSH+ACK. Overall, PSH elicited more responses, but PSH+ACK elicited larger ones.

SYN with Payload This strategy sends the forbidden GET request as the payload of a single SYN packet. This elicited the fewest responses—21 (11.4%) of the middleboxes—but one of the largest amplification factors: 527×.

It is not common to send payloads in SYN packets[4], which led us to hypothesize that the middleboxes that responded to this might *only* be looking at the payloads. But this appears not to be the case: only 3 (14.3%) of the middleboxes that responded to SYN also responded to PSH+ACK, and only 6 (28.6%) also responded to PSH.

3.3.2 Packet Sequence Modifications

Geneva identified five additional modifications to the above packet sequences that improve the amplification factor for some middleboxes. One of these (increasing TTLs) never resulted in lower amplifications, and appear to be worth doing against all middleboxes. Four improve amplification for some middleboxes but lower it for others; to use such modifications in a practical setting, an attacker would ideally identify the middleboxes it uses ahead of time.

Increased TTLs Every IP header includes a time-to-live (TTL) field to limit the number of hops a packet should take; routers are supposed to decrement this at each hop, and drop the packet if the TTL reaches zero. Against one middlebox, Geneva learned to increase the TTL of both packets in the ⟨SYN; PSH+ACK⟩ sequence to its maximum value (255) to improve the amplification factor. It is very surprising that the TTL would have any impact on the amplification factor; the default TTL was already large enough to reach the destination.

To understand its root cause, we sent packet sequences to this middlebox with TTLs ranging from 0 to 255, and counted the number of responses for each. We find a perfectly linear relationship between TTL and amplification factor: we received $t-13$ block pages for all TTL values $t \geq 13$. At the maximum TTL value (255), it sent 242 copies of its block page!

This behavior can be explained by *routing loops* in the network of the censoring middlebox. Each time the packet sequence circles the routing loop, it re-crosses the censoring middlebox, causing it to re-inject its block page. That this only works for TTLs greater than 13 indicates that the routing loop is 13 hops from our measurement host. We show in §5

[4]This is generally reserved for TCP Fast Open, which is rare in practice.

that routing loops are surprisingly common on the Internet at large, and they can be exploited by attackers for significant improvements to the amplification factor.

We found that setting a high TTL on packets has no effect on the response rate of any of the other packet sequences, so this modification can be made at no cost to freely exploit routing loops for maximum amplification.

Increased `wscale` Window scaling (or `wscale`) is a TCP option that controls how large the TCP window can grow. Geneva discovered an optimization that gets 7 (3.8%) more middleboxes to respond to the ⟨SYN; PSH+ACK⟩ sequence: setting the `wscale` TCP option in the SYN packet to an integer greater than 12. Based on the block page these middleboxes injected, we believe they are instances of Symantec's Web Gateway (SWG).

To understand this behavior, we sent the modified packet sequence 1,000 times to the candidate middleboxes in Quack's dataset, and repeated this experiment five times. Strangely, in each case, the middleboxes responded only ∼25% of the time. We could successfully ping the end-hosts behind each SWG with innocuous requests, suggesting that packet drops are not the root cause of the reduced response rate. Varying the time between each packet sequence had no effect on the response rate, indicating we were not overloading the SWGs. The behavior is also not affected by packets sent by the end-host: if we limit the TTL of all of our packets such that they reach the middlebox but not the end-host, the middlebox still injects content to 25% of requests. Finally, altering the actual value of `wscale` had no effect on response rate. We do not understand why SWG is sensitive to this option.

Like with increased TTLs, increasing `wscale` had no adverse effect on response rates or sizes. However, because `wscale` is a TCP option, it requires additional bytes, thereby potentially lowering the amplification factor.

TCP Segmentation One modification Geneva identified for some middleboxes is to simply segment the forbidden GET request across multiple packets, either by adding an additional packet to single-packet sequences, or across the two packets in the ⟨SYN; PSH⟩ or ⟨SYN; PSH+ACK⟩ sequences. Geneva discovered that 5/184 (2%) middleboxes would send the block page a second time, once for each packet segment. For these middleboxes, this serves as an optimization for the amplification factor: although it comes at the cost of an additional packet with some payload, the payoff is a doubling in traffic elicited from the middleboxes. Strangely, this modification only works for two segments: any further segmentation causes two of the middleboxes to not respond, and the other three only send a maximum of two block pages.

Although this optimization can improve the amplification from middleboxes with this behavior, 26 others (14%) are unable to perform packet reassembly and stop responding entirely. Worse, for the middleboxes that do perform reassembly and still respond, segmenting the request across multiple packets lowers the amplification factor.

`FIN+CWR` Another modification Geneva identified against four (2%) middleboxes was to change the TCP flags of the PSH+ACK packet in the ⟨SYN; PSH+ACK⟩ sequence to FIN+CWR. The CWR flag—"Congestion Window Reduced"—is used for TCP's Explicit Congestion Notification (ECN), and generally should not be combined with a FIN flag. The modified packet sequence elicits 12 copies of the middleboxes' block pages, each sent 0.4 seconds apart. The block page duplication increases the amplification factor of these middleboxes to 301×. If the CWR flag is not present on the packet, no response is sent. According to the injected block pages, these middleboxes appear to be instances of Fortinet Application Guard; this modification appears to only improve amplification factor for these middleboxes.

Shorter HTTP Geneva discovered an optimization against one middlebox: cutting off the four bytes in the HTTP GET request that immediately follow the forbidden URL (`\r\n\r\n`). Although this slightly improves the amplification factor for one middlebox, none of the other 183 middleboxes responded. This suggests that it is important for the HTTP GET request to be well-formed.

Failed Approaches We expected that changing the TCP `window` in our packet sequences might have an impact on amplification. Recall that TCP `window` size determines how much data the other endpoint can send before expecting an acknowledgement. However, we found that none of the middleboxes respected this TCP feature. Similarly, though TCP mandates that data sent should not exceed the maximum segment size (MSS) TCP option, every middlebox ignored this option.

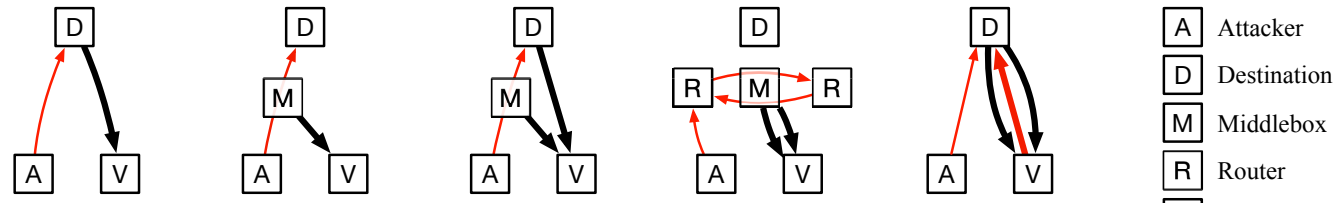
4 Internet Scanning Methodology

We perform ZMap [10] scans of the IPv4 Internet to measure the effectiveness each of the attack packet sequences from §3.

Modifications to ZMap ZMap allows us to create arbitrary probe packets with the "probe modules"; we wrote a custom probe module for the packet sequences identified by Geneva. ZMap does not natively have the ability to send multiple distinct packets in each probe (e.g., SYN followed by PSH+ACK), so we modified ZMap to add this capability.

Selecting Forbidden URLs Quack's dataset contains 1,052 URLs that triggered censorship. Ideally, we could perform full Internet-wide scans for *each* URL and determine which ones produce the highest amplification. Unfortunately, this would take over 6 weeks of scanning at full 1 Gbps line rate per Geneva strategy, and would likely have diminishing returns.

Instead, we chose to estimate the smallest combination of URLs that collectively elicit responses from the largest number of IP addresses. To do this, we construct every set of



(a) Destination reflection (b) Middlebox reflection (c) Destination and middlebox reflection (d) Routing loop reflection (e) Victim-sustained reflection

Figure 3: Types of attacks we find. Thick arrows denote amplification; red ones denote packets that trigger amplification. We find that infinite amplification is caused by (d) routing loops that fail to decrement TTLs and (e) victim-sustained reflection.

size $1 \leq N \leq 7$ of the 1,052 URLs from the Quack dataset, and for each set compute the number of Quack IP addresses it would have triggered.

We find the ideal set to be of size $N = 5$, each coincidentally from a different website category as identified by the Citizen Lab Block List [17]: www.youporn.com (pornography), plus.google.com (social networking), www.bittorrent.com (file sharing), www.roxypalace.com (online gambling), and www.survive.org.uk (sexual health services). These five keywords collectively elicit responses from 83% of the Quack IP addresses, after which there are diminishing returns (adding a sixth keyword only increased the response rate by 3.6%).

We acknowledge that the Quack dataset may not be representative of the entire Internet. Moreover, coverage of IP addresses is not necessarily the same as coverage of middleboxes; however, few IP addresses (4%) in the Quack dataset share the same /24 prefix, so we expect little middlebox overlap. It is possible that other keywords will elicit broader coverage or greater amplification; we leave this to future work.

Data Collection From April 9th to April 26th, 2020, we performed 5 sets of Internet scans, one for each mutually exclusive packet configuration (§3.3). For each set, we performed 7 Internet-wide scans: one for each of the 5 domains and our two control scans ("`example.com`", and no payload at all). To avoid saturating our link, we scanned at 350 Mbps; and each scan took approximately 2–4 hours. After each scan, we aggregated the number of bytes and packets we received from each IP address that responded to our probes. Following convention, we include the size of the Ethernet header in the size of our probes and response packets when computing amplification factors.

5 Internet Scanning Results

This section presents the results of sending our attack packet sequences from §3 to the entire IPv4 Internet. We make two notes upfront that are important in understanding our results:

Responder variation Our packet sequences elicit a wide range of behaviors. We broadly classify them in Figure 3;

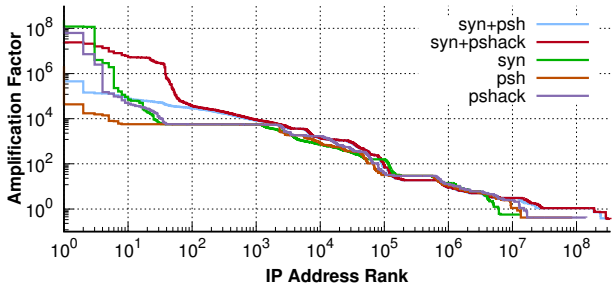


Figure 4: Rank order plot of the amplification factor received from each IP address for the triggering payloads containing `www.youporn.com` across all five packet sequences.

for some destinations and packet sequences, we get response packets directly from destinations, from middleboxes (pretending to be the destination), or some combination of the two. We confirm in §5.3 that over 82% of the largest responses we receive come from middleboxes, but unfortunately it is difficult to perform this analysis for *every* destination IP address we send to. Thus, for consistency (and because middlebox dealiasing is difficult and error-prone), we report on the number of destination IP addresses from which we can elicit responses throughout this paper. We explore clustering and identifying middleboxes by their responses in §5.4.

Infinite amplification We discover many IP addresses that continue to respond, seemingly indefinitely, to our probes. The amplification factors for these IP addresses are technically *infinite*, but we report the (finite) amplification we obtained during our scans. These tend to be orders of magnitude larger than other hosts. We explore infinite amplifiers in §6.

5.1 Which strategies work best?

We begin by measuring the impact that packet sequence and keyword have on response rate and amplification factor.

Figure 4 compares the amplification factors for each of the 5 packet sequences with the URL `www.youporn.com`. We immediately observe that each of these strategies elicits responses from over 5M destination IP addresses with

URL	SYN	PSH	PSH+ACK	⟨SYN; PSH⟩	⟨SYN; PSH+ACK⟩
www.youporn.com	49.4	4.4	23.2	13.9	**52.0**
roxypalace.com	5.8	4.4	16.5	13.6	**31.3**
plus.google.com	7.4	7.0	5.9	13.4	**14.9**
bittorrent.com	3.7	3.2	3.8	10.6	**13.7**
survive.org.uk	4.4	2.8	2.4	11.0	**11.2**
example.com	3.4	2.9	2.8	**11.2**	8.4
empty	**0.06**	0.01	0.02	0.05	0.06

Table 2: Total data received (GB) from the top 100,000 IP addresses for each combination of target URL and packet sequence. Bolded is the maximum value *for each target URL*.

URL	SYN	PSH	PSH+ACK	⟨SYN; PSH⟩	⟨SYN; PSH+ACK⟩
www.youporn.com	116,120	**67,503**	**78,830**	**92,765**	97,689
roxypalace.com	**128,843**	52,168	63,080	86,010	97,213
plus.google.com	39,177	27,815	24,827	54,916	63,090
bittorrent.com	33,187	19,171	24,682	47,348	**193,754**
survive.org.uk	98,038	14,600	13,060	45,953	43,927
example.com	28,909	15,669	15,911	46,469	27,962
empty	65	27	49	42	59

Table 3: Number of IP addresses with amplification factor over 100× for each combination of target URL and packet sequence. Bolded is the maximum value *for each sequence*.

amplification greater than one. Moreover, we find that all of them elicit very large amplification factors; for each packet sequence, there are over 50,000 destination IP addresses that yield over 100×.

To focus on the heaviest hitters, Table 2 compares the total volume of traffic generated from the top 100,000 IP addresses for each scan, and Table 3 shows the number of IP addresses with amplification factor greater than 100×. ⟨SYN; PSH⟩ and ⟨SYN; PSH+ACK⟩ get responses from the largest number of unique IP addresses: 29× more than the SYN scan. Despite requiring an additional packet, they also yield higher amplification factors for most of the top 1,000 IP addresses, and elicited the highest total amount of traffic across every URL. Sending a SYN packet with a forbidden HTTP GET was surprisingly effective at eliciting responses: for half of the URLs, it had the most IP addresses with an amplification factor greater than 100×.

The choice of URL has a strong impact on how well a given packet sequence amplifies. Figure 5 shows the amplification factors from using each of the keyword/strategy combination.

Overall, `www.youporn.com` was the most effective for eliciting the most responses, with two notable exceptions. First, `www.bittorrent.com` elicited double the number of IP addresses with amplification factor greater than 100×. The source of this is highly amplifying censorship of two networks with /16 prefixes: one run by the University of Ghent; the other, the City of Jacksonville, Florida. Second, `roxypalace.com` on SYN packets similarly elicited responses from more IP addresses than any other URL, and

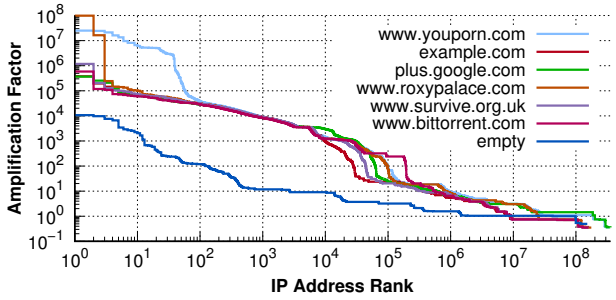


Figure 5: Rank order plot of the amplification factor received from each IP address for the ⟨SYN; PSH+ACK⟩ packet sequence across all seven scanning payloads.

this is largely due to triggering the border firewall at Brigham Young University, which runs a /16 prefix.

Surprisingly, scans for the control keyword `example.com` trigger many amplifiers. It under-performed every other keyword in number of IP addresses and amount of data elicited, but thousands of IP addresses still responded with 20× amplification. It is possible the middleboxes who respond to this do so as a means of access control. Scans with an empty payload received the fewest amplifiers, smallest total data elicited, and smallest total amplification: the ⟨SYN; PSH+ACK⟩ scan elicited *three orders of magnitude* more data than an empty SYN scan.

Summary The ⟨SYN; PSH+ACK⟩ packet sequence with `www.youporn.com` is overall the most effective at eliciting amplification, but other URLs and sequences are needed to trigger specific, large networks.

5.2 Are these actually amplifiers?

We next explore if these IP addresses can be (ab)used for real-world attacks. In a real attack, an attacker would not send just one trigger packet sequence; instead, she would repeatedly send trigger packet sequences to these IP addresses to amplify the response traffic. To test if the IP addresses we identify are true amplifiers, we perform an experiment with the top 1 million IP addresses with the highest amplification factor from the ⟨SYN; PSH+ACK⟩ scan with `www.youporn.com` keyword. Using ZMap, we perform two independent scans to these IP addresses: first, by sending 5 trigger packet sequences to each IP address, and second (as a control), just one trigger packet sequence[5].

Figure 6 presents the *increase factor*: the ratio of bytes we received from each IP address when sending 5 probes to the bytes received from 1 probe. Perfect amplifiers have an increase factor of 5×. Our results suggest that the majority of the top 1 million IP addresses are true amplifiers. Over 46% of IP addresses responded with exactly 5× as much data, and

[5]When sending multiple probes, we modify ZMap so that each probe is sent from a different source port, so the packets are not identical.

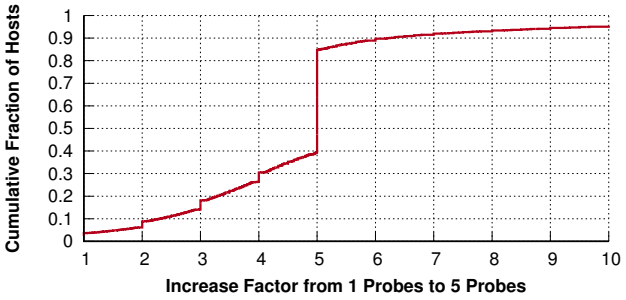


Figure 6: The increase factor in the number of bytes we receive between sending 5 probes and sending 1 probe. 46% of IP addresses responded with exactly 5× as much data.

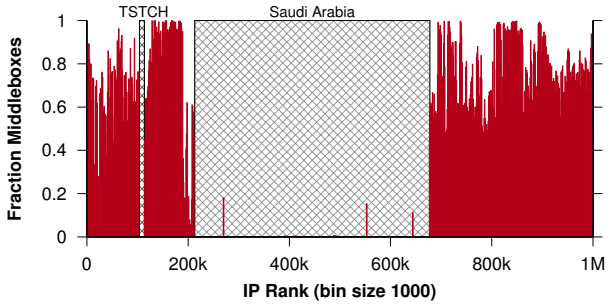


Figure 7: The fraction of the top million hosts that we confirm are middleboxes, using TTL-limited probe. The small gap at $x \approx 100{,}000$ and the large gap in the middle of the plot correspond to networks that block traceroutes at their borders. Accounting for this, we find injected responses from 82.9% of the top million IP addresses are from confirmed middleboxes.

another 30% responded with between 2× and 5× as much data, likely representing amplifiers that missed or dropped one or more of our packets. Notably, many of the IP addresses that sent the most data do not increase by the same rate. Of the top 100 amplifiers, none of them increased by exactly a factor of 5×, and only 10 increased by 4–6×.

5.3 Are these middleboxes?

Next, we determine if the responses we receive are truly coming from middleboxes. We performed a traceroute using a custom ZMap probe module on the top million IP addresses by bytes received in our ⟨SYN; PSH+ACK⟩ www.youporn.com scan. Our ZMap module sent three TTL-limited TCP SYN packets for each TTL between 10 and 25 to each of the million hosts, and recorded the resulting ICMP TTL-exceeded messages. This allowed us to construct a (partial) traceroute for each target for hops 10–25. Out of the million targets, 99.5% provided at least one router hop, with an average of at least 6 hops per traceroute.

For each target, we extracted the last hop that we received a TTL-exceeded message for (i.e., the last hop we learned on the traceroute to the target). We then sent a follow up ⟨SYN; PSH+ACK⟩ sequence with www.youporn.com to the target, but TTL-limited to the last known hop. This probe is certain to not reach the target, as it should generate a TTL-exceeded message by the last-hop router. Therefore, if we still receive a response from the endpoint, we can tell the response is coming from a middlebox along the path to the target, and not the target itself.

If we do not receive a response, we cannot conclude that responses normally come from the target endpoint, as it could be that our traceroute was incomplete: there may be a middlebox further along the path but still before the endpoint. However, we can interpret the presence of a response to our TTL-limited probe as confirmation that it was produced by a middlebox.

Figure 7 shows the results of this scan, binning IP addresses into bins of size 1,000 and plotting the fraction of the IPs in the bin that we identified as middleboxes. Overall, 36.8% of the 1M targets responded to our TTL-limited probe, positively confirming their responses were produced by a middlebox. Notably present, however, are two gaps in the graph in which almost no responses were received:

The small gap has ∼10,000 IP addresses ($104{,}000 \leq x \leq 114{,}000$). All of these IPs are in three /20-sized subnets that belong to the Texas State Technical College Harlingen (TSTCH). Their responses correspond to block pages generated by a SonicWall network security appliance, a common middlebox we see in our data. It appears that TSTCH blocks traceroutes at its border, meaning that our last-observed traceroute hop occurs before the SonicWall appliance.

The larger gap has ∼465,000 IP addresses ($213{,}000 \leq x \leq 678{,}000$). 98.6% of them geolocate to Saudi Arabia. Looking at their traceroutes, their last hops comprise just 2,068 unique router IPs, with 90% of IP addresses sharing only 10 last-hop routers (all within Saudi Arabia). It appears that Saudi Arabia also blocks traceroutes at their border, preventing us from being able to traceroute into the country. However, the response that comes back from 97% of the IP addresses in this block corresponds to the standard block page of Saudi Arabian censorship, describing that the website is blocked, and also suggesting a middlebox is responsible for this response.

Conservatively labelling the 10,000 IP addresses from TSTCH and 97% of the 465,000 Saudi Arabian IPs as encountering on-path middleboxes increases the percent of IPs that encounter on-path middleboxes to 82.9% of the million targets we scanned. We conclude that responses from the vast majority of IP addresses in our dataset are produced by middleboxes.

5.4 What kind of packets do amplifiers send?

We analyzed the packets we received in our ⟨SYN; PSH+ACK⟩ scan with www.youporn.com. This scan received a total of

Country	#Responsive IP addresses	% Sending fingerprint	Fingerprint
China	170,858,209	90.0%	3× RST+ACK (54)
S Korea	15,981,100	7.6%	PSH+FIN+ACK (119)
Iran	8,612,544	75.7%	PSH+FIN+ACK (402–405); RST+PSH+ACK (54)
Egypt	2,909,897	89.8%	RST+ACK (54)
Bangladesh	1,375,908	81.4%	PSH+FIN+ACK (248)
Saudi Arabia	894,858	45.3%	PSH+ACK (97); 2× PSH+ACK (1354)
Oman	596,546	94.7%	RST (54)
Qatar	387,625	89.4%	RST (54)
Uzbekistan	253,098	91.8%	FIN+ACK (74)
Kuwait	173,126	31.3%	PSH+FIN+ACK (114)
UAE	161,014	52.0%	RST (54)

Table 4: Nation-states with nation-wide censorship infrastructure and the fingerprint they most frequently respond to clients with. Numbers in parentheses denote packet sizes in bytes.

over 105 GB of data from 337 million IP addresses. For each IP address, we generate a *fingerprint* from the response packet sequence, consisting of a vector of `(TCP flags, packet size)` tuples; this allows us to efficiently group IP addresses that send us similar responses. We then counted the number of IP addresses that sent each fingerprint. We ignore order to allow for packet re-ordering.

Overall, we discover **63,662 unique fingerprints**. Each fingerprint represents a unique set of packets sent by amplifiers. The fingerprint returned by the most IP addresses is a sequence of three 54-byte RST+ACKs, which we received from approximately 154 million IPs. This is a well-known censorship pattern produced by the Great Firewall of China (GFW) [6, 42], and using the MaxMind database [21], we find 99.9% of these IPs geolocate to China. We note this is weakly-amplifying, sending 162 bytes for our 149 byte probe.

The fingerprints representing the largest number of bytes are less common. For example, the top fingerprint is 528,007 410 byte FIN+PSH+ACK packets and 525,110 RST+ACKs, sent by a single IP address in India. We investigate these mega-amplifiers more in §6. The largest fingerprints sent by more than one IP address consist of a single SYN+ACK and multiple megabytes worth of PSH+ACK packets containing data. These appear to be sent by buggy TCP servers that simply respond to our non-compliant GET request with real data. We find approximately 746,000 IP addresses with this behavior.

5.5 Are these national firewalls?

We find that nation-state censorship infrastructure makes up a significant fraction of the TCP amplifiers we discover. Figure 8 breaks down the amplification we see for the top 5 countries by number of amplifying IP addresses. Out of these, all but the US have deployed nationwide Internet censorship infrastructure [11, 12], visible by long flat plateaus in the graph which indicate a large number of IP addresses with uniform amplification. The US is a notable exception, and

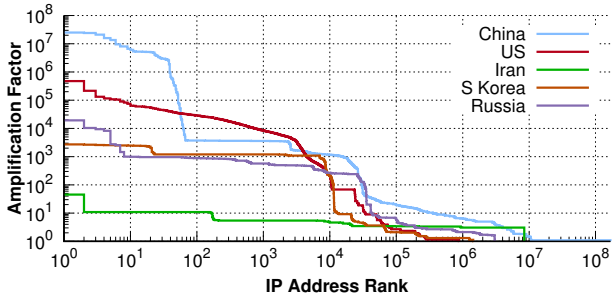


Figure 8: Rank order plot of the amplification factor by country for the `www.youporn.com` scan with the ⟨SYN; PSH+ACK⟩ packet sequence.

we explore why it is so prevalent later in this section. Amplification factors vary significantly country-to-country due to different censorship methods.

By extracting fingerprints that were shared by many IP addresses that geolocate to the same country, we can identify censoring nation-states. For example, over a million IP addresses geolocate to Bangladesh and respond with a 248-byte FIN+PSH+ACK. Table 4 shows a sample of censoring countries and their most popular fingerprint. At a slightly higher amplification, we observe four similar fingerprints with two packets each: a 402–405-byte FIN+PSH+ACK and a 54-byte RST+PSH+ACK. We received these fingerprints from 8.6 million IP addresses in Iran, representing 76% of all the responding IP addresses that geolocate to Iran.

The censorship infrastructure of Saudi Arabia also shows prominently in our dataset: its fingerprint is three packets: a 97-byte PSH+ACK and two 1354-byte PSH+ACKs, offering an amplification factor of 18.9×. We received this fingerprint from over 400K IP addresses, 99% of which geolocate to Saudi Arabia, comprising 45% of all the responding IP addresses that geolocate to Saudi Arabia.

In general, we find the amplification factor from nation-state censors is small: most countries we surveyed provide less than 4× amplification. The GFW of China is the largest—but also the weakest—amplifier we find. Curiously, we find that the GFW has a *different fingerprint* between two of our scans: the ⟨SYN; PSH+ACK⟩ scan with `plus.google.com` elicited three RST+ACKs and a RST packet, but this extra RST packet is missing in scans for `www.youporn.com`. This RST was also absent when `plus.google.com` was sent with the ⟨SYN; PSH⟩ sequence. The presence of the RST raises the amplification factor of the GFW from 1.08× to 1.45×.

We do not understand why the GFW behaves differently between these keywords and sequences. Researchers have hypothesized that the RST+ACK and RST packets from the GFW originate from different, co-located censorship systems [6,42]; our results support this theory, and even suggest that the block lists themselves can be processed differently between the two censorship systems depending on the sequences of packets.

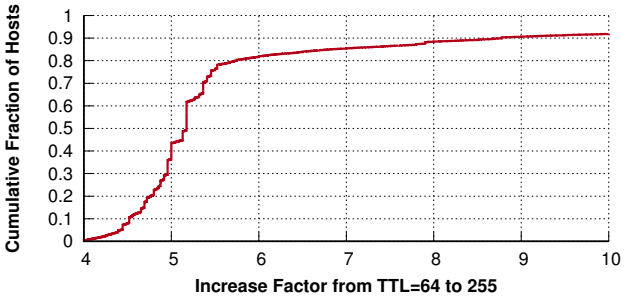


Figure 9: CDF of the increase factor in amplification of candidate looping IP addresses when scanned with a TTL of 255 and 64. Because the increase factor is affected by the number of hops away an IP address is, we expect routing loops to have an increase factor of at least 4. Larger increase factors are further away from our scanner, limiting the overall amplification factor from our perspective.

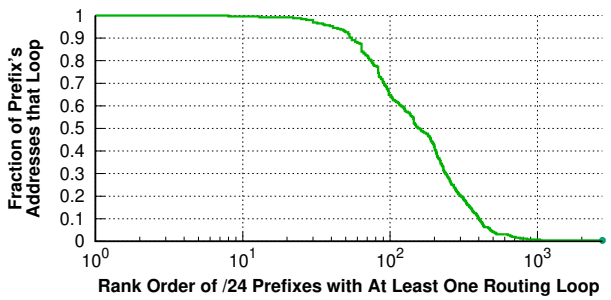


Figure 10: The /24 prefixes with at least one routing loop, rank-ordered by the fraction of their 256 IP addresses that we observe to loop. Of the 2,763 looping prefixes, 54 (2%) have over 90% of their IP addresses loop, but 1,705 (62%) have only one looping IP address. (Note that the x-axis is log-scale.)

We also discover hundreds of IP addresses in routing loops in Russia that contain censoring middleboxes with 250.9× amplification. The highest amplifying nation-state censors are two censoring ISPs located in Russia that seem to have *infinite* routing loops in their network, that sent us packets for weeks after our scans. We examine the effects of routing loops more closely next in §5.6.

Nation-state censors pose a more significant threat to the Internet than their amplification factor alone suggests. First, nation-state censorship infrastructure is located at high-speed ISPs, and is capable of sending and injecting data at incredibly high bandwidths. This allows an attacker to amplify larger amounts of traffic without worry of amplifier saturation. Second, the enormous pool of source IP addresses that can be used to trigger amplification attacks makes it difficult for victims to simply block a handful of reflectors [29]. Nation-state censors effectively turn every routable IP addresses within their country into a potential amplifier.

While nation-state censors are well-represented in our amplifiers dataset, other large non-censoring countries, such as the US, are prevalent as well. Specifically for the US, we observe a more diverse set of fingerprints: over 13,000 unique fingerprints, compared to 7,553 in Russia, and under 3,000 from South Korea. This indicates a diversity of networks, rather than a coordinated, nationwide deployment. Indeed, we observe several university and enterprise firewalls that respond with identifiable and amplifying fingerprints.

These results demonstrate that nation-state censors enable TCP amplification attacks, but that they are far from the sole contributor to this problem.

5.6 Routing Loops

Routing loops are the result of network misconfigurations, inconsistencies, and errors in routing protocol implementations. Packets caught in a routing loop will typically eventually be dropped when their TTL reaches zero. However, even a finite routing loop can hypothetically have significant impact on amplification factor. Suppose an amplifying middlebox were in a routing loop; every time an offending packet traversed the loop, it would re-trigger the middlebox. Such a scenario would make the network self-amplifying: at no additional cost to an attacker, the effective amplification rate of a middlebox would be increased by the number of times the packet crosses the middlebox in the routing loop.

The maximum value of TTL in the IPv4 header is 255, so the number of times a single trigger packet sequence can elicit responses from an RFC-compliant middlebox is $\ell(255-d)$, where d is the number of hops between the attacker machine and the routing loop and ℓ is the number of times the packets traverse the amplifying middlebox per loop.

So far, our scans were conducted with a TTL value of 255, in accordance with the optimizations discovered by Geneva in §3. We performed follow-up scans with a reduced TTL value in order to observe which IP addresses send us a corresponding reduction in the number of packets, allowing us to identify which amplifiers involve routing loops.

For this experiment, we use the ⟨SYN; PSH+ACK⟩ packet sequence with the `www.youporn.com` trigger keyword. We use the top 1 million hosts (by number packets sent during the scans), and perform two follow-up scans to these IP addresses: one with the TTL set to 255 and one set to 64 (approximately 1/4 the value). As we are knowingly re-triggering machines with potentially enormous amplification factors, we reduced the scanning speed to 100 kbps[6].

We can identify routing loops by comparing the number of packets we receive per IP address across scans. For a routing loop d hops from our scanner, we expect a probe with TTL = 255 to receive $(255-d)/(64-d)$ times more packets than

[6]Despite our low send rate, we received back on average around 800 Mbps, representing a total amplification of 8,000× for this experiment.

a probe with TTL = 64. Note that this value increases as d increases, and, for a routing loop, has a minimum value of ∼4 (when the routing loop is zero hops away). Therefore, we label an IP addresses as having a routing loop if it has an increase factor of at least 4 and sent more than 10 packets when probed with a TTL of 255. From our top 1 million IP sample, we label **53,041 IP addresses as routing loop amplifiers** using this heuristic, spanning 2,763 distinct /24 prefixes. Figure 9 presents a CDF of the increase factor for these routing loop IPs.

Loops per subnet One would expect that if sending to a given IP address results in a routing loop, then all of the other IP addresses in its /24 prefix would experience a loop, as well. Surprisingly, we find that 62% of /24 prefixes with at least one routing loop have *exactly* one loop. Figure 10 shows the fraction of IP addresses found in each looping /24 prefix. Only 54 subnets have over 90% (231 of 256) of their IP addresses show evidence of being a routing-loop amplifier. On the other hand, 81.2% (2,244) of looping prefixes have fewer than 10 looping IP addresses. This means that even if an attacker can elicit responses from a middlebox by sending packets to any IP address that routes through it, she may only be able to take advantage of routing loops to a small number of IP addresses.

6 "Mega-amplifiers"

In our scans, we identify a surprising number of hosts that send enormous amounts of data in response to a single packet sequence—on the order of many gigabytes. We believe these are the same "mega-amplifiers" that Czyz et al. [9] reported in 2014. We identify two phenomena that contribute to mega-amplification: self-sustaining amplifiers and victim-sustained amplifiers.

Self-Sustaining Amplifiers *Self-sustaining* amplifiers are IP addresses that, once triggered, continue sending data indefinitely. In our scans, we have observed these continuing for *weeks* after our probes. We hypothesize the cause of self-sustaining amplifiers is infinite routing loops: routing loops between middleboxes that do not decrement TTLs.

An infinite routing loop suggests these amplifiers are sending responses at the maximum capacity of their links. To confirm, we sent a packet sequence to a self-sustaining amplifier we identified in an ISP's censorship system in Russia. A short time later, we sent the same packet sequence from a different vantage point, and we recorded the bandwidth received from each. Figure 11 shows the bandwidth we received on both vantage points during our experiment. When we send a probe from a second vantage point, the response bandwidth was split equally between them.

We were unable to terminate the barrage of packets sent to us by this amplifier. We sent RST packets, and also tried FIN+ACK, FIN, RST+ACK, and ICMP port unreachable messages with no effect. Ultimately, the traffic stopped after approximately six days to the first vantage point, and 22 hours for the second. We believe the reason they finally stopped was because the routing loop eventually dropped a packet.

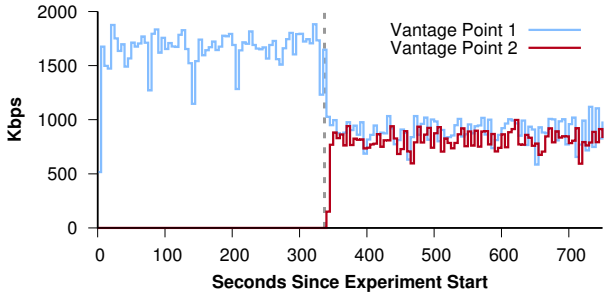


Figure 11: Attack bandwidth received at two vantage points from a self-sustaining amplifying IP address, which (based on its block page) appears to be a component of a Russian ISP's censorship system. The dashed line marks when the packet sequence was sent from the second vantage point. Note how the bandwidth we get from the system is divided evenly between the vantage points. This experiment supports our hypothesis that self-sustaining amplification is caused by an infinite routing loop.

Fortunately, we find very few self-sustaining amplifiers: only 19 IP addresses sent data continuously. We identified 6 IP addresses (each in a different /24 prefix) located in China that sent the known censorship pattern from the GFW indefinitely, possibly indicating a loop across the GFW itself. Two ISPs in Russia also sent block pages indefinitely.

Victim-Sustained Attacks The TCP standard says that when a host receives an unsolicited non-RST packet, it should send a RST packet in response [28]. For TCP amplification victims, this means they will send RST packets for any received (amplified) traffic. Normally, victim-generated RST packets have no effect on middlebox amplifiers[7].

However, our scans identify amplifying IP addresses that send an *additional response* to RST packets instead of ignoring them. This causes the victim to send another RST, inducing more responses, and so on. This packet storm continues indefinitely until a packet is dropped.

By default, our scanning machine sent outbound RST packets in response to data, thereby eliciting additional packets from victim-sustained amplifiers. To explore the effect that outbound RST packets have on amplification factor, we perform two additional scans: one with outbound RST packets turned off for the www.youporn.com keyword in the ⟨SYN; PSH+ACK⟩ sequence, and one with RSTs enabled (default). Figure 12 shows a comparison between these two scans. Dropping outbound RST packets has the effect of lowering the amplification factor for the top amplifying IP addresses,

[7] Conversely, they may serendipitously halt SYN-based amplification attacks that target end-hosts [15, 16].

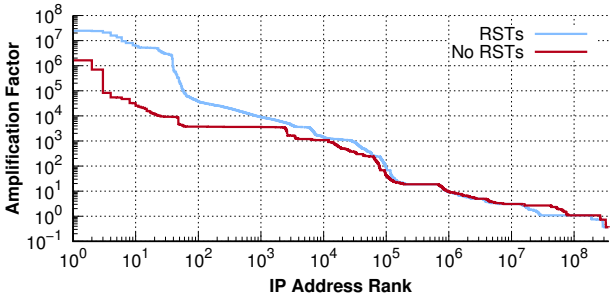


Figure 12: Rank order plot of amplification factor of two scans for the `www.youporn.com` keyword requested with the ⟨`SYN`; `PSH+ACK`⟩ packet sequence: one with outbound `RST` and `RST+ACK` packets being dropped and the other normally.

while raising the amplification factor of many IP addresses in the "long-tail".

We find several thousand IP addresses that behave this way, which we classify into two classes: censoring repeaters and "acknowledgers".

For **censoring repeaters**, we find 4,154 middleboxes that re-send a block page in response to a `RST`. This appears to be a buggy flow-tracking middlebox that, once a TCP flow triggers blocking, will continue injecting its block page in response to *any* subsequent packet, including `RST`s.

For **acknowledgers**, we find 10,645 IPs that respond with an `ACK` to both data payloads and subsequent `RST` packets. This behavior is also not TCP compliant. To investigate what operating systems these "acknowledgers" are, we performed Operating System (OS) identification `nmap` [19] scans on 500 randomly sampled victim sustained IP addresses. Of the 452 (90.2%) IP addresses with a successful OS match, 267 (59%) were Dell SonicWall NSA 220. We believe this firewall model is to blame for most of the acknowledger victim-sustained behavior: the next most common OS match was Linux 2.6[8], with only 14 hosts (3%).

7 Ethical Considerations

Internet Scanning We followed best practices for scans as outlined by ZMap and Quack [10, 41]. We set up reverse DNS and hosted a webpage on the IP address we performed scans from, explaining the purpose of our scans. We also listed an email address to receive complaints and allow people to opt out of future scans. We received 8 removal requests over the course of our study comprising 2.1 million IP addresses which we removed from our scans.

Censorship-focused Internet-wide scans require additional careful considerations to avoid causing harm or falsely implicating users in making censored requests. In prior work on active probing to trigger censorship, researchers used alternative techniques to avoid having clients in censored countries make requests for banned content [11, 25, 34, 41]. Similarly in our work, the requests are made by our scanning machine from outside the censored countries to all IPv4 addresses, making it unlikely that a government would punish any individual, due to the directionality and ubiquity of the scans. The packet sequences we probe with are non-TCP compliant and do not induce any in-country clients to make sensitive requests in response. For these reasons, we believe wide-scale scans of this nature pose minimal risk to individuals in censored regions.

Saturation Experiments A natural question with all amplification studies is: at what point do amplifiers' link saturate? For example, a single host with amplification factor of 5,000× may not be very valuable if it only has a 100kbps uplink.

Measuring the saturation of a specific amplifier requires sending the triggering packet sequence in rapid succession and measuring the response it triggers. For ethical reasons, we do not perform such an experiment. These experiments would effectively perform denial of service attacks against the specific middlebox or the IP address, or could adversely impact other networks on path.

We unintentionally triggered mega-amplifiers, and report on our findings in this paper. However, after discovering these IP addresses and the nature of their responses, we removed them from future scans.

Responsible Disclosure Responsibly disclosing our findings is challenging given the large number of potentially affected vendors and network operators. It is both difficult to fingerprint specific vendors or manufacturers of middleboxes, and also difficult to identify the networks where middleboxes are responding from, as they spoof their source IP address by design.

Nonetheless, we attempted to reach out to both operators and vendors of middleboxes we discovered in our study. We contacted several country-level Computer Emergency Readiness Teams (CERT) that coordinate disclosure for their respective countries, including China, Egypt, India, Iran, Oman, Qatar, Russia, Saudi Arabia, South Korea, the United Arab Emirates, and the United States. We also reached out to several middlebox vendors and manufacturers, including Check Point, Cisco, F5, Fortinet, Juniper, Netscout, Palo Alto, SonicWall, and Sucuri.

We also publicly provide a repository of scripts that can help manufacturers and network operators test their middleboxes for amplifying behavior.

8 Related Work

TCP Reflected Amplification Attacks In 2014, Kürher et al. introduced a TCP handshake amplification attack [15, 16] that takes advantage of a server retransmitting `SYN+ACK` packets multiple times in response to a single `SYN`. They find

[8] We note this is not standard Linux 2.6 behavior.

millions of hosts that will retransmit up to 20×, though most send fewer than 6. We also observe this attack in our work, but additionally discover hundreds of millions more IPs with orders of magnitude higher amplification rates.

Non-reflective Amplification Attacks Other amplification attacks abuse TCP but involve directly connecting to the victim. Sherwood et al. [36] showed an attacker can use *optimistic acknowledgments* to induce a server to send a file at higher rates, ultimately DoSing *its own network*. The Great Cannon injects Javascript into Baidu webpages, turning visiting browsers into denial of service bots [20]. Our attack is effectively the reverse: instead of a censor co-opting the bandwidth of users to perform an attack, an attacker can co-opt the bandwidth of the censor.

UDP Reflected Amplification Attacks Reflected UDP attacks have been studied extensively [1, 18, 32, 37]. However, we are the first to study the use of middleboxes as reflectors.

Victim-sustained Attacks Sargent et al. [33] identified 79 hosts that respond to a particular IGMP request by repeating the request. Ostensibly, source-spoofing this request could cause an infinite loop between two such hosts, and is thus similar to our victim-sustained attacks in §6. Our attacks are more widely applicable, since they rely on standard client behavior (sending RSTs to unsolicited packets); and as a result we identified several orders of magnitude more targets of victim-sustained infinite amplification. However, their findings motivate applying tools like Geneva at the *application layer* to discover application-specific bugs.

9 Countermeasures

Unlike previous amplification attack vectors [9, 32, 37], our attack is not isolated to a specific protocol and impacts a wide range of implementations and devices. Unfortunately, this means there is no single vendor or network that can be patched to correct the problem. Instead, this issue is systemic to middleboxes, particularly those that must operate seeing only one side of a connection.

Nonetheless, we offer potential remedies that can eliminate or partially mitigate amplification attacks, for both middleboxes and potential victims.

9.1 Middleboxes

Connection directionality While many middleboxes see asymmetric sides of a connection (e.g., only traffic to the server), there are others that see both sides, such as middleboxes deployed at the gateways of networks. These middleboxes can accurately infer if a connection is live and only inject content if the three-way handshake is valid. We recommend such middleboxes require seeing traffic in both directions (to client and to server), and only inject block pages if this condition is met. This makes it more difficult for an attacker to spoof a connection, as it is infeasible for them to get both sides of a spoofed connection to pass by the same middlebox to induce injection. However, this solution will not work for large-scale middleboxes that sit in large transit networks and more frequently see only one side of a connection.

Limit injected response sizes Some middleboxes inject large block pages, directly enabling large amplification attacks. An alternative approach is for these middleboxes to only respond with a single RST to close a forbidden connection, or a with a minimal HTTP redirect to a different server that hosts a block page. If the middlebox's response size is smaller than the minimum size required to trigger it, this ensures that the middlebox will not be a productive amplifier.

Egress filtering Though middleboxes are only supposed to block websites for a limited group (such as a country or within a corporate or school network), many operate "bidirectionally", such that users outside the network accessing content within can also trigger injected responses. For instance, users outside China can still elicit the Great Firewall of China to inject RST packets despite not being the intended target of censorship. Instead, middleboxes should be configured to only censor requests originating from within the intended network, limiting the scope of victims of amplification.

Remove or limit censorship devices Many middleboxes inject block pages into censored HTTP requests which use an outdated protocol that has been far surpassed in traffic volume and page loads by HTTPS [38]. The utility that HTTP-injecting devices provide is shrinking, and will ultimately disappear as more sites use TLS. However, the damage they inflict via amplification attacks will remain until these devices are removed. Disabling HTTP injection in these devices altogether would prevent abuse from attackers.

9.2 End Hosts

End hosts can take steps to mitigate the potential impact of these attacks. Hosts that drop outbound RST packets are more susceptible to TCP handshake-based attacks, but hosts that do not are susceptible to sustaining a packet storm from a victim-sustained amplifier. Instead, we recommend end hosts be configured to drop outbound RST packets probabilistically; this prevents an infinite packet storm, while still offering some protection from handshake-based amplifiers.

10 Conclusion

We presented the first non-trivial TCP-based reflected amplification attacks. To discover them, we made use of a novel genetic algorithm that we trained directly against censoring middleboxes. We then scanned the Internet dozens of times

and find over 200 million IPv4 addresses that provide amplification from 1× to over 700,000×, as well as others that effectively yield *infinite* amplification.

Through a series of thorough follow-up experiments, we found that these TCP amplifiers are predominantly middleboxes, and frequently nation-state censorship devices. It has long been understood that nation-state censors restrict open communication for those in their borders; our work shows that they pose an even greater threat to the Internet as a whole, as attackers can weaponize their powerful infrastructures to attack anyone.

Our results show that middleboxes introduce an unexpected, as-yet untapped threat that attackers could leverage to launch powerful DoS attacks. Protecting the Internet from these threats will require concerted effort from many middlebox manufacturers and operators. To assist in these efforts, we have made our code publicly available at:

https://geneva.cs.umd.edu/weaponizing

Acknowledgments

We thank the network infrastructure team at the University of Colorado Boulder for supporting our scanning efforts and providing the resources that made this work possible. We also thank the anonymous reviewers for their helpful feedback. Finally, we thank our collaborators from the OTF and OONI communities for contributing resources that enabled this work. This research was supported in part by the Open Technology Fund and NSF grants CNS-1816802 and CNS-1943240.

References

[1] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. DNS Amplification Attack Revisited. *Computers & Security*, 39(B):475–485, November 2013.

[2] Bilal Anwer, Theophilus Benson, Nick Feamster, and Dave Levin. Programming Slick Network Functions. In *Symposium on SDR Research (SOSR)*, 2015.

[3] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet Censorship in Iran: A First Look. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[4] Robert Beverly and Steven Bauer. The Spoofer Project: inferring the Extent of Source Address Filtering on the Internet. In *USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.

[5] Kevin Bock, Yair Fax, Jasraj Singh, Kyle Reese, and Dave Levin. Iran: A New Model for Censorship. https://geneva.cs.umd.edu/posts/iran-whitelister.

[6] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. Geneva: Evolving Censorship Evasion Strategies. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[7] Eric Cronin, Micah Sherr, and Matthew A. Blaze. The Eavesdropper's Dilemma. Technical report, Penn Engineering, February 2006.

[8] CVE-2018-1000115: Memcached version 1.5.5. National Vulnerability Database, March 2018. http://nvd.nist.gov/nvd.cfm?cvename=CVE-2018-1000115.

[9] Jakub Czyz, Michael Kallitsis, Manaf Gharaibeh, Christos Papadopoulos, Michael Bailey, and Manish Karir. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *ACM Internet Measurement Conference (IMC)*, 2014.

[10] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. ZMap: Fast Internet-wide Scanning and its Security Applications. In *USENIX Security Symposium*, 2013.

[11] Arturo Filasto and Jacob Appelbaum. OONI: Open Observatory of Network Interference. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[12] Freedom House. Freedom in the world report. https://freedomhouse.org/countries/freedom-world/scores.

[13] Ben Jones, Tzu-Wen Lee, Nick Feamster, and Phillipa Gill. Automated Detection and Fingerprinting of Censorship Block Pages. In *ACM Internet Measurement Conference (IMC)*, 2014.

[14] Sam Kottler. February 28th DDoS incident report. https://github.blog/2018-03-01-ddos-incident-report/, Mar 2018.

[15] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *USENIX Security Symposium*, 2014.

[16] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2014.

[17] Citizen Lab. Block test list. https://github.com/citizenlab/test-lists.

[18] Bingshuang Liu, Skyler Berg, Jun Li, Tao Wei, Chao Zhang, and Xinhui Han. The Store-and-Flood Distributed Reflective Denial of Service Attack. In *International Conference on Computer Communication and Networks (ICCCN)*, 2014.

[19] Gordon Lyon. nmap. https://nmap.org/.

[20] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. An Analysis of China's "Great Cannon". In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2015.

[21] MaxMind. GeoLite2. https://dev.maxmind.com/geoip/geoip2/geolite2, 2020.

[22] Allison McDonald, Matthew Bernhard, Luke Valenta, Benjamin VanderSloot, Will Scott, Nick Sullivan, J. Alex Halderman, and Roya Ensafi. 403 Forbidden: A Global View of CDN Geoblocking. In *ACM Internet Measurement Conference (IMC)*, 2018.

[23] Robert T. Morris. A Weakness in the 4.2BSD Unix TCP/IP Software. CSTR 117, 1985.

[24] Zubair Nabi. The Anatomy of Web Censorship in Pakistan. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[25] Craig Partridge and Mark Allman. Addressing ethical considerations in network measurement papers. In *NS Ethics@ SIGCOMM*, 2015.

[26] Vern Paxson. End-to-End Routing Behavior in the Internet. In *ACM SIGCOMM*, 1996.

[27] Censored Planet. Censored Planet Raw Data. https://censoredplanet.org/data/raw.

[28] John Postel. Transmission control protocol. RFC 793, Internet Engineering Task Force, September 1981.

[29] Matthew Prince. The ddos that almost broke the internet. Cloudflare Blog, March 2013. https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/.

[30] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. Censored Planet: An Internet-wide, Longitudinal Censorship Observatory. In *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[31] Ram Sundara Raman, Adrian Stoll, Jakub Dalek, Armin Sarabi, Reethika Ramesh, Will Scott, and Roya Ensafi. Measuring the Deployment of Network Censorship Filters at Global Scale. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[32] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[33] Matthew Sargent, John Kristoff, Vern Paxson, and Mark Allman. On the Potential Abuse of IGMP. *ACM SIGCOMM Computer Communication Review (CCR)*, 47(1), 2017.

[34] Will Scott, Thomas Anderson, Tadayoshi Kohno, and Arvind Krishnamurthy. Satellite: Joint Analysis of CDNs and Network-Level Interference. In *USENIX Annual Technical Conference*, 2016.

[35] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service. In *ACM SIGCOMM*, 2012.

[36] Rob Sherwood, Bobby Bhattacharjee, and Ryan Braud. Misbehaving TCP Receivers Can Cause Internet-Wide Congestion Collapse. In *ACM Conference on Computer and Communications Security (CCS)*, 2005.

[37] Kulvinder Singh and Ajit Singh. Memcached DDoS Exploits: Operations, Vulnerabilities, Preventions and Mitigations. In *International Conference on Computing, Communication and Security (ICCCS)*, 2018.

[38] Let's Encrypt Stats. Percentage of Web Pages Loaded by Firefox Using HTTPS. https://letsencrypt.org/stats/#percent-pageloads, 2018.

[39] The Spoofer Project: State of IP Spoofing. https://spoofer.caida.org/summary.php.

[40] UDP-Based Amplification Attacks: Alert (TA14-017A). National Cyber Awareness System Alerts, January 2014. https://www.us-cert.gov/ncas/alerts/TA14-017A.

[41] Benjamin VanderSloot, Allison McDonald, Will Scott, J. Alex Halderman, and Roya Ensafi. Quack: Scalable Remote Measurement of Application-Layer Censorship. In *USENIX Security Symposium*, 2018.

[42] Zhongjie Wang, Yue Cao, Zhiyun Qian, Chengyu Song, and Srikanth V. Krishnamurthy. Your State is Not Mine: A Closer Look at Evading Stateful Internet Censorship. In *ACM Internet Measurement Conference (IMC)*, 2017.

[43] Zhongjie Wang, Shitong Zhu, Yue Cao, Zhiyun Qian, Chengyu Song, Srikanth V. Krishnamurthy, Kevin S. Chan, and Tracy D. Braun. SymTCP: Eluding Stateful Deep Packet Inspection with Automated Discrepancy Discovery. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[44] Xing Xu, Yurong Jiang, Tobias Flach, Ethan Katz-Bassett, David Choffnes, and Ramesh Govindan. Investigating Transparent Web Proxies in Cellular Networks. In *Passive and Active Network Measurement Conference (PAM)*, 2015.

[45] Xueyang Xu, Morley Mao, and J. Alex Halderman. Internet Censorship in China: Where Does the Filtering Occur? In *Passive and Active Network Measurement Conference (PAM)*, 2011.

[46] Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. Where The Light Gets In: Analyzing Web Censorship Mechanisms in India. In *ACM Internet Measurement Conference (IMC)*, 2018.