An Inter-domain Collaboration Scheme to Remedy DDoS Attacks in Computer Networks

Steven Simpson, Syed Noorulhassan Shirazi, Angelos Marnerides *Member, IEEE*, Simon Jouet, Dimitrios Pezaros *Senior Member, IEEE*, David Hutchison

***Abstract*—Distributed Denial-of-Service (DDoS) attacks continue to trouble network operators and service providers, and with increasing intensity. Effective response to DDoS can be slow (because of manual diagnosis and interaction) and potentially self-defeating (as indiscriminate filtering accomplishes a likely goal of the attacker), and this is the result of the discrepancy between the service provider's flow-based, application-level view of traffic and the network operator's packet-based, network-level view and limited functionality. Furthermore, a network required to take action may be in an Autonomous System (AS) several AS-hops away from the service, so it has no direct relationship with the service on whose behalf it acts. This paper presents *Antidose*, a means of interaction between a vulnerable peripheral service and an indirectly related AS that allows the AS to confidently deploy local filtering with discrimination under the control of the remote service. We implement the core filtering mechanism of Antidose, and provide an analysis of it to demonstrate that conscious attacks against the mechanism will not expose the AS to additional attacks. We present a performance evaluation to show that the mechanism is operationally feasible in the emerging trend of operators' willingness to increase the programmability of their hardware with SDN technologies such as OpenFlow, as well as to act to mitigate attacks on downstream customers.**

***Index Terms*—Distributed Denial-of-Service, Antidose, mitigation, BPFabric, network security, network resilience, bandwidth saturation attacks, network management, inter-domain collaboration**

I. Introduction

In a bandwidth-saturating Distributed Denial-of-Service (DDoS) attack, thousands or even millions of malicious network hosts, usually compromised machines of unsuspecting users, conspire to flood a target host or network with such high volumes of traffic that legitimate users are unable to access services hosted there. Links and queues outside the target network but leading to it can be saturated by traffic, leaving the target network inaccessible remotely, regardless of its local capacity. Such attacks could be classified according to [1] as VT-4 (Network attacks) and IV-1:PDR-1 (Disruptive; Self-recoverable).

DDoS attacks are of a simple yet very effective class [2], but their impact in recent decades has been significant. These attacks can generate traffic in the order of hundreds of Gbit/s (e.g., on Github [3] and the BBC [4]), possibly through the use of DDoS-for-hire services also known as booters [5]. In 2016, the largest ever DDoS attack was recorded, exceeding 1 Tbit/s, along with increased complexity and ease of deployment by means of IoT devices, impacting organizations including many running critical services [6]. Such incidents can translate into millions of dollars of lost revenue, yet DDoS defense remains an open research issue [7].

Having detected that some target[1] is under attack, mitigation of its effects remains challenging because the vulnerability of the attack (a link's capacity) and the target are not necessarily in the same administrative domain, i.e., Autonomous System (AS). Flows containing attack traffic must be filtered before their aggregates exceed downstream link capacity, but ASes commanding these locations lack a means to accurately determine whether a packet is good or bad as soon as it arrives. Meanwhile, the target may have a sufficiently detailed view to discriminate accurately, but does not command the filtering locations in potentially remote ASes. If the target could express its discriminator to sufficiently upstream ASes, malicious packets could be dropped before their flows coalesce, while letting good packets pass.

Figure 1 shows attack flows coalescing towards target T. At some point, the aggregated volume of these flows exceeds the capacity of the links they traverse. Network components downstream of such links are in a *saturation zone*, where inbound capacity is exhausted, and this zone has a *saturation boundary*. An attacker can draw the boundary away from the target by attacking in higher volume, rendering local DDoS defense mechanisms ineffective, and requiring co-operation across administrative domains. The further the saturation boundary from the target, the harder it is for the necessary actors to take action, as the operators of network components at the boundary are more distinct from the target service providers. Being in distinct ASes, these actors can be unfamiliar with each other, having no prior trust or formal relationships (such as SLAs or peering agreements if they are more than one administrative hop away), so they may be unwilling to co-operate to mitigate the attack. If an AS on the saturation boundary is to act on behalf of the target, it must be confident that the request to act is genuine (i.e., not from some malicious actor pretending to be the target), and that it

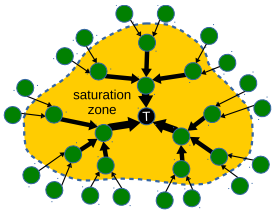

Fig. 1. Saturation zone and boundary

S. Simpson, S. Shirazi, A. Marnerides and D. Hutchison are with the School of Computing and Communications, Lancaster University, LA1 4WA, UK.
S. Jouet and D. Pezaros are with the School of Computing Science, University of Glasgow, G12 8QQ, Scotland.

[1]We use 'target' to refer to the definitive parameter of an attack, in contrast to 'victim', an entity affected negatively by the attack, which usually includes the target.

is taking the correct action to deal with the attack (blocking enough attack traffic, while allowing enough legitimate traffic through). Furthermore, ASes (especially those closest to the core) may be reluctant to bear the overhead of implementing a sophisticated filter on hardware optimized for simple, high-speed destination-based forwarding. The filtering overhead and memory requirements must therefore be minimized, and shown to be feasible in a high-performance AS context.

A discriminator must be carefully selected. 'All traffic to T' will accomplish the attacker's likely goal (to make T inaccessible), but could be used in combination with source-address filtering. Blacklists of source IP addresses may be inappropriate within a network if spoofing is present or the number of malicious addresses is large. The Spoofer project[2] shows that, at the start of 2017, close to a fifth of Internet addresses, and a quarter of autonomous systems, allow their hosts to spoof. The analysis of backscatter presented in [8] also suggests that, despite the proliferation of NAT devices, spoofing is still widespread, and the next generation of attacks may intelligently probe networks and adapt their behavior based on the ability to spoof [9], [10]. When attacks involve NTP or DNS(SEC) reflection, source spoofing is used to engineer them, but the packets that contribute to saturation do not have spoofed source addresses, suggesting that spoofing need not be considered during mitigation. However, without additional context, it is very difficult for the ASes serving the target to determine whether any given packet has a spoofed source address. As a result, the success of a mitigation mechanism that does not take account of this uncertainty could motivate attackers to use spoofing more directly, or populate the mechanism's blacklist with innocent parties, or over-populate it. As a discriminator, a *whitelist* of all potential legitimate addresses would be absurd. A whitelist of just the subset with which the target *presently* communicates is much more feasible, and could be populated only with addresses demonstrated to be unspoofed, but poses a considerable barrier to new legitimate clients [11].

Inside an AS, the discriminator must be able to operate on individual packets, not just on flows or other higher-level constructs (ASes see only the network level), without expensive analysis or additional queuing delays. Packets cannot be allowed to be queued or otherwise stored in the AS to any greater extent than they would under normal circumstances, as this presents more vulnerable volume-based resources to the attacker, as well as degrading QoS for the end users.

The deeper into the network the saturation boundary penetrates, the more the network is geared to simple, high-speed operation, and the fewer are the opportunities to inject programming into packet forwarding. The discriminator must be simple enough to be efficiently implemented within an AS. Filtering at line rate requires much computation and bespoke programmable hardware operating at performances well beyond the practical limits of software-based systems [12]–[14].

Finally, any requirement to additionally interact with legitimate peers of the target should be minimized, as any remedy requiring new forms of interaction with the peer will suffer from inertia. In particular, techniques that require *all* peers to take part are unlikely ever to be adopted.

This paper presents the design, specification and implementation of a DDoS remedy *Antidose*, and we report on its adaptation to the *BPFabric* environment, a high-performance programmable data-plane fabric described in [15]. Antidose defines a discriminator based on a whitelist of known good peers without barring entry of new ones, whether they or the target initiate new interactions. The discriminator can be propagated across ASes, and is simple enough to be implemented as C compiled to eBPF, a platform- and target-independent instruction set designed for real-time packet processing, which an AS can then deploy. By deploying the discriminator in the BPFabric software switch environment, we determine its ability to accurately separate target-identified 'good' packets from other traffic. We contend that Antidose's ability to discriminate even in restricted environments such as BPFabric demonstrates that automatic remedial collaboration between networks within and at the edge of the saturation zone is a feasible and practical proposition. (BPFabric can also exploit the zero-copy packet-processing infrastructure DPDK[3] and has the potential to exploit greater hardware acceleration.) Antidose is active only temporarily, i.e., during an attack, and only in ASes in and at the edge of the saturation zone.

This paper makes the following contributions: motivates and proposes Antidose; specifies the format and computation of proofs and cookies (the essential elements of Antidose), and the interaction of ASes to exchange them; implements the main Antidose component as a data-plane function to be deployed as part of network switching fabric; and demonstrates the impact of Antidose on target-identified 'good' traffic versus other traffic, and shows that false positives ('other' traffic mistakenly allowed to pass) remain minimal even when many distinct 'good' flows are identified.

Section II discusses the nature of the most recent and worst DDoS attacks. Section III specifies Antidose as a collection of data structures, actors and their interactive behavior. Section IV describes our implementation of Antidose. Section V analyzes the functional behavior of Antidose, discusses potential attack strategies that could defeat it, and evaluates its performance under some such conditions. Section VI discusses limitations and remaining challenges of Antidose, including some practicalities of deployment. Section VII relates Antidose to prior work, and Section VIII concludes the paper.

II. Background

Fundamental aspects of the Internet facilitate DoS attacks and hamper their mitigation [16]. Also, the expansion of the Internet in recent decades has presented more potential victims, more devices to attack from/with, and more motivation to attack. It is also easy to launch a DoS, whereas the cost of victimhood or remedy is high. On the 21st of October 2016, the Internet infrastructure company Dyn suffered the largest DDoS attack to date, which resulted in inaccessibility of many services such as Twitter, GitHub, PayPal, Etsy and others.

[2]https://www.caida.org/projects/spoofer/

[3]Data Plane Development Kit http://dpdk.org/

While details of how the attack happened remain vague, it is certain that the Internet is fragile in the face of DDoS attacks.

An empirical study of DoS attacks observed 68 000 directed at over 34 000 distinct victim IP addresses over roughly the years 2001 to 2003 [17]. These attacks targeted Amazon, Hotmail and ISPs. They were later analyzed by an independent technology research firm[4] and it was observed that they had significant financial impact on companies, estimated at $1.2 billion in revenue during the observed period [18]. To determine the prevalence of DoS attacks, understand their nature, and see long-term trends, we collected information about DoS attacks reported in the SANS Newsbites news feed[5]. It is clear from these anecdotal reports that DoS attacks distributed among many different domains and ISPs are prevalent, and such attacks are a common threat for sites depending on the Internet. However, there is little quantitative data about these attacks, as operators consider such information private or sensitive. It is also evident that there is an increasing trend of these attacks in terms of frequency and intensity. During 2015, the largest DDoS attack thus far was recorded at 400–500 Gbit/s, and another at the beginning of 2016 exceeded it with 602 Gbit/s[6]. The Dyn attack of October 2016 was reported[7] to have up to 100 000 malicious endpoints and an (unconfirmed) magnitude of 1.2 Tbit/s. DDoS mitigation giant Akamai released in their quarterly security report[8] (Q1-2015) that there was a 116.5 % increase in total DDoS attacks, a 59.83 % increase in application-layer DDoS attacks, a 124.69 % increase in infrastructure (layer 3 and 4) DDoS attacks, and a 42.8 % increase in the average attack duration, compared to Q1-2014.

Based on these reports, it is evident that volumetric (bandwidth-saturation) attacks are not likely to abate while the vulnerability continues to exist. The emergence of 'booters', essentially commercial DDoS service providers, will also increase the options available to even novices with malicious intent. A remediation strategy is required that network operators find feasible and inexpensive to implement, that they can confidently co-operate on, and that does not indiscriminately cut off all service to the attack target.

III. Design

Antidose is a collection of packet formats, protocols and functions that an AS may choose to implement to various degrees such that it can counter a DDoS attack on a target it has no direct knowledge of. Additional functions are required at the target, and optionally at its peers if they initiate communication with the target (i.e., it is a server, and they are its clients). We define *target agents* and *client agents* as the entities implementing these functions, and they need not be integrated with the target/client, just sufficiently co-located. A single client agent could also serve multiple co-located clients.

[4]https://en.wikipedia.org/wiki/Yankee_Group

[5]https://www.sans.org/newsletters/newsbites/newsbites.php

[6]http://www.zdnet.com/article/attackers-targeting-bbc-donald-trump-amazon-web-services/

[7]http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

[8]https://www.akamai.com/uk/en/about/news/press/2015-press/akamai-state-of-the-internet-security-report.jsp

Antidose should be engaged only when the target agent determines that it is necessary, and is to be disengaged when the target agent deems the attack to be over. At all other times, an AS should behave as normal. Temporary engagement of Antidose is in line with the Remediate and Recover steps of the $D^2R^2 + DR$ strategy [19], whereby a normally suboptimal network configuration is temporarily adopted during a challenge because it prevents total loss of service.

A target agent directly asks only its hosting AS to engage Antidose. That AS may ask its immediate neighbor ASes to engage, and they will ask theirs in turn, etc. For the highest levels of conformance to Antidose, this engagement only needs to propagate just beyond the saturation boundary.

Antidose defines a portable packet discriminator that a target agent can maintain control over remotely while participating ASes apply it. In essence, the discriminator is a whitelist of legitimate IP addresses (and specific port numbers if necessary) currently interacting with the target, and governs filters selectively deployed within any collaborating AS. Legitimate clients make an effort (partially through their agents) to get themselves into the whitelist, but only under their own IPs. Although an attacker could spoof a whitelisted peer, attacking hosts are not generally expected to know the specific set of current legitimate peers; except by chance, traffic with a spoofed source address should not get through a filter.

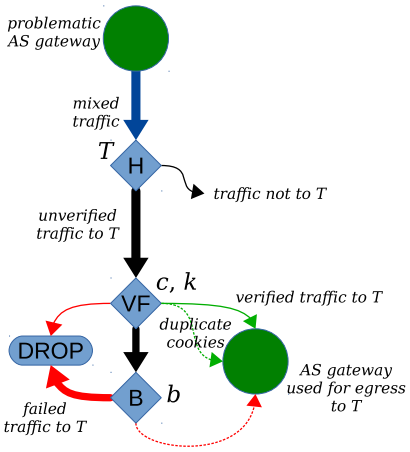

Fig. 2. Separation of verified traffic from unverified in a single AS

Whitelist membership decays over time, and must be periodically refreshed. Although an attacker could initially behave well enough to be whitelisted, it could then only attack using its real address, risking exposure of its bad behavior on that address to the target agent, which could respond by locally blacklisting the address, and refusing to refresh it remotely.

An AS implementing Antidose for a given target T selects filtering points within its network, and ensures that all traffic to T goes through at least one such point. Filtering points must be outside the saturation zone if they are to prevent attack flows from coalescing enough to saturate a downstream link; otherwise, when the AS is wholly within the zone, it may still select filtering points locally, but it will also require an upstream AS to act on its behalf.

A *verification filter* (VF) exists at each filtering point, and contains the whitelist and other state pertaining to T. A VF checks packets heading to the target, and decides to let them pass if the source address is in the whitelist, or to throttle them (i.e., to pipe them through a more restricted channel) if not.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2828938, IEEE Transactions on Network and Service Management

Figure 2 depicts the flow of traffic through a filtering point in an AS. The H component simply selects only traffic going to T, allowing the rest to be forwarded in the usual manner. B is a bandwidth restriction which will allow a small amount b of unverified traffic through.

Whitelists within different VFs need not have the same content. A given VF's whitelist only needs to contain a subset of peer IPs whose traffic passes through that VF, so no synchronization between VFs' whitelists' membership is required.

Whitelists are updated remotely by the transmission of *flow cookies* (FCs; Section III-A) from the target agent. Clients (and their agents) act to persuade the target agent to whitelist them, but cannot reliably interact with the target until whitelisted, so a second mechanism, *proof-of-work* (p/w; Section III-B), is employed to get critical initial packets from the client through the filters, which can then persuade the target agent to whitelist the client. In cases where the target normally initiates legitimate interactions (e.g., it is a DNS client flooded with misdirected DNS responses in a DNSSEC UDP reflection attack), this mechanism is redundant, as the target simply issues cookies to peers it knows it is about to interact with. In Figure 2, VF is configured with c (a proof-of-work challenge) and k (a public key to verify flow cookies) provided by T.

Flow cookies:

An Antidose flow cookie is a cryptographic statement certifying that a given IP address (and optionally a port) is permitted to send to a target. The target agent generates cookies for a client after observing that the client is behaving well with an unspoofed address. It withholds cookies if a client subsequently appears to be behaving badly, allowing the client's membership to decay from the relevant whitelists.

A cookie is delivered by encapsulating it in an ICMP Echo Request[9] (a ping) from the target to the client. Assuming the client (agent) echoes it back, verification filters will observe it as an ICMP Echo Reply from client to target. The filter is separately and previously provided with the credentials necessary to verify the cookie, and adds the sender address to the whitelist, then forwards the cookie as normal, if it checks out. If not, the filter can safely drop it.

An asymmetric cypher is required, as verification credentials could reach the attacker in the worst case. Asymmetric cryptography incurs a performance cost (sometimes greater in signature verification than in signing [20]), although only signature verification is required inside the verification filter. (Signing is done only by the target agent.)

Proof-of-work:

A proof of work is the solution to a hash-function challenge. The challenge for a solver is to identify a solution such that a hash function over various parameters including the solution yields a hash code with a bit pattern matching another in certain bits. The hash function must be irreversible to ensure that solutions can only be obtained through brute force.

A client agent attaches proofs to packets as a form of per-packet authorization. With access to the same parameters, a VF can verify that a proof is correct by performing the same hash computation. Unlike the solver, which must perform multiple computations to find a useful solution, the verifier has to perform only one. Proof-of-work therefore places a significant processing burden on agents of all clients (legitimate or not), but little burden on in-network filters.

Some parameters identify the flow to which the packet carrying the proof belongs. Others are specified by the target agent, and can be adjusted to make the challenge more difficult, without increasing the load on the verifier. Parameters can be safely distributed to attackers, as this merely allows them to perform verification, an operation which is of no use to them.

The verification filter retains valid cookies and proofs that it has already seen, so they cannot be reused. Timestamps in cookies and proofs allow a filter to discard old ones, and retain only the most recent ones, requiring only roughly synchronized clocks. Greater asynchrony can be tolerated in exchange for larger or less accurate cookie/proof sets.

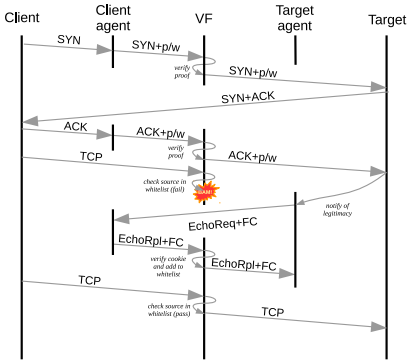

Fig. 3. Interaction between new client, server and intervening verification filter (VF)

Figure 3 shows how a new legitimate client interacts with a target server and an intervening VF in forming a TCP connection. Its initial packets are tagged by its agent with unique and recently computed proofs-of-work, which allow the SYN and first ACK to pass through a VF unhindered. Responses from the server are not observed by VF, so always pass. On reception of the first ACK, the target can be confident that the client is using its real address, and notifies its agent[10] that the client should be whitelisted, triggering the periodic emission of flow cookies to the client. When these return and pass through the VF, it verifies them, and adds or reinforces the client's identity in the whitelist. Until that first addition, other packets from the client that do not carry unique proofs will follow the restricted path from the VF, and will likely be dropped.

At the start of an attack, a target server may also preemptively issue cookies to its current or most recent clients, obviating their sending of proofs.

[9] Alternatives are not precluded, but ICMP Echo requires no changes in the client.

[10] Alternatively, the agent observes the handshake, and so infers legitimacy.

A. Flow cookies

A flow cookie is a non-repudiable statement made by one peer permitting another to send to it. A cookie is formed by signing a byte structure consisting of the following information: a timestamp in seconds; a serial number identifying cypher and key; the IP protocol number of the protocol that the cookie permits; a weight for increasing the presence of the peer in a whitelist; a list of port numbers (empty to mean all; 16-bit unsigned integers); the IP address of the permitted peer (the subject host); and the IP address of the permitting peer.

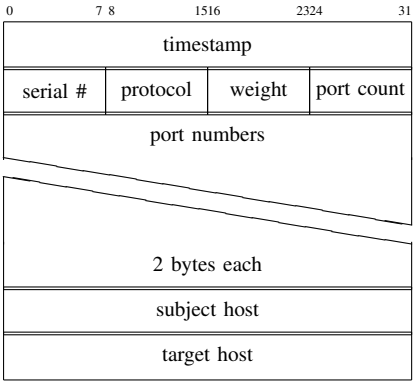

Fig. 4. Byte sequence for computing flow-cookie signatures

Figure 4 shows how these fields are encoded as a byte sequence, using network byte ordering.

A cookie itself is formed from the same byte sequence, except that the host fields are removed, a magic word is prefixed, and the signature is suffixed, as shown in Figure 5. This structure is then encapsulated in an ICMP Echo Request, and issued by (or on behalf of) the target host, destined for the subject host.

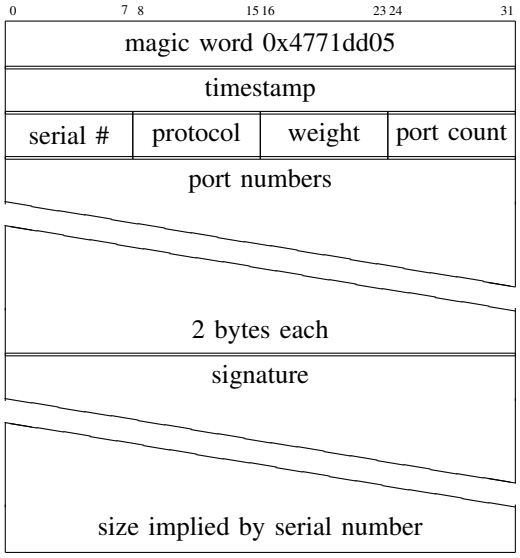

Fig. 5. Flow cookie as ICMP Echo payload

The subject host having echoed the cookie back, a VF readily identifies an ICMP Echo Reply with a payload beginning with the magic word as a potential flow cookie. The serial number identifies the the cypher and key (provided to the VF through another channel) to be used to check the signature. The subject host is now implicitly the sender of the packet, while the target host is the destination.

B. Proof-of-work

The proof-of-work computation involves an irreversible hash function H over a tuple of several parameters and a candidate solution S. If $\mathrm{bin}(H(\langle S, t, h, a, F, T, l\rangle)) \cap \mathrm{bin}(2^l - 1) = \emptyset$, then S is valid[11]. t is a timestamp in seconds; a is the IP protocol; F is the 'from' address, i.e., the client solving the challenge; T is the 'to' address, which is (part of) the target of the attack. h is a seed and l is a difficulty level, both specified by the target agent. The candidate solution S is always first in the tuple input to the hash function, so that an attacker cannot use a partial hash state computed for one solution to check another.

h can be replaced if the target agent is concerned that it has somehow been compromised. The agent might also suspect that the attacker has a large amount of processing capability at its disposal, and can respond by increasing l to make the challenge harder. Of course, it will then be harder for all unwhitelisted clients to find valid proofs, but it will be no harder for verifiers. Also, each proof only allows one packet through; a couple of proofs attached to a legitimate client's initial packets can buy it membership in the whitelist, obviating further proofs, while attackers who do not wish to expose their true identities only get one packet through per average unit of effort.

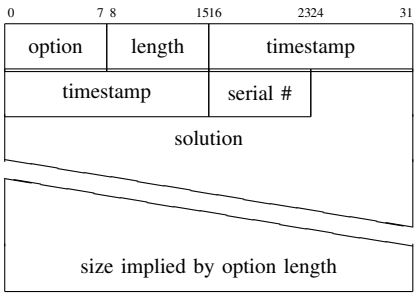

Fig. 6. Proof-of-work as IP option

For the purposes of hashing and transmission, the timestamp t is a 4-byte big-endian integer (as before). F and T are 4 or 16 bytes of the respective IP addresses, plus 2 bytes (big-endian) for the port number if the encapsulating protocol supports the concept. h and S are arbitrary-length byte sequences, though practical limits may be imposed.

Figure 6 shows a proof-of-work encapsulated as an IP option. Other means of attaching a proof to a packet are not precluded.

C. Verification filter (VF)

Figure 7 shows the behavior of a verification filter. It has a single 'INPUT', and four outputs ('DROP', 'PASS', 'PRIORITY' and 'THROTTLE') reflecting the four decisions it can reach for each packet (cf. Figure 2). It also retains three state entities:

- The cookie set contains *valid cookies* that have already been seen, to prevent replay attacks. Duplicate valid cookies are not necessarily dropped, as they might serve to whitelist a client in a downstream VF. Indeed, duplicates for a whitelisted address are given higher priority over other traffic. Invalid or broken (including expired) cookies

[11] In other words, the bottom l bits of the hashcode must be zero.

can be safely discarded, as they will not serve any purpose to a downstream VF.

- The whitelist is a set of source tuples (e.g., a combination of IP address and protocols, and optionally port number) of clients permitted to send to the target. Valid cookies cause the presence of their source tuples to be reinforced in the whitelist. Presence in the whitelist decays over time, so new cookies for already present tuples must be seen to retain a tuple indefinitely.
- The proof set contains *valid proofs* that have already been seen, to prevent replay attacks. As with cookies, valid duplicate proofs are allowed to pass, but as throttled traffic. Invalid ones are always dropped, as it is expected that they would be dropped in downstream VFs too.

Note that proofs do not influence the whitelist, so providing a proof will not directly get a potential client whitelisted. Proof verification comes after whitelist checking, as it only needs to be performed on unwhitelisted traffic. Whitelisting is performed after cookie verification, as a cookie must still be processed whether it would pass the whitelist or not. Reversing the order (i.e., checking the whitelist before verifying the cookie) merely results in duplicating the cookie-verification step on alternative flow branches, and offers no protection against attempts to overload cookie verification.

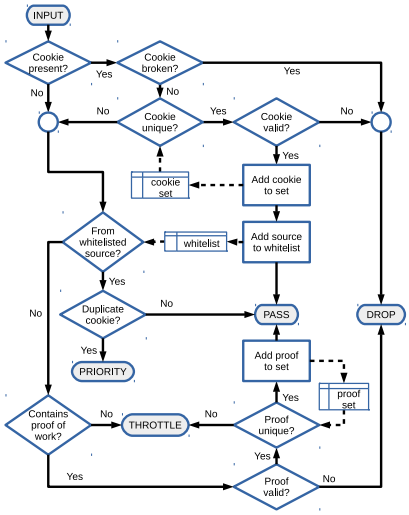

Fig. 7. Verification filter (VF) packet flow

D. Infrastructure

To operate effectively, Antidose must be activated in at least the ASes on the saturation boundary. ASes exchange Antidose signalling information so that they can correctly set up and configure VFs when they are needed, and deactivate them after an attack. Antidose signalling primarily conveys the parameters c, k and b, the proof-of-work challenge, the flow-cookie credentials and the bandwidth restriction respectively. c itself consists of $\langle h, l \rangle$, the seed and difficulty of a proof-of-work challenge, and also includes the hash-function type. k consists of the public key corresponding to the private key used to sign cookies, as well as type information for the cypher and any necessary hash functions. c and k are themselves elements of series of parameters with monotonically increasing serial numbers, allowing them to be replaced and updated if compromised.

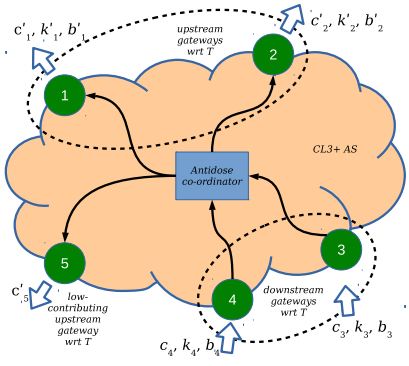

Fig. 8. Propagation of p/w and FC credentials across AS

An AS partaking in Antidose is doing so on behalf of the target (as well as of collateral victims, including itself), so it must have confidence that the request to act is genuine and will serve the target. For this reason, when the AS and target have no direct trust relationship, they do not communicate directly. Instead, the target's agent communicates with the *Antidose co-ordinator* of its provider AS, which communicates with the co-ordinators of its immediate neighbor ASes, which communicate with their neighbors, and so on. This also has the advantage of allowing channels to be reserved or prioritized for Antidose communication between neighbor ASes, vital if the rest of an AS's bandwidth is consumed by attack traffic. Figure 8 illustrates an AS with a co-ordinator receiving Antidose parameters for target T from 'downstream' gateways through which it currently forwards traffic to T. Having distilled this information, it passes some of it on to other 'upstream' gateways that are not currently used to deliver to T. Simultaneously, the co-ordinator can use the information to establish, configure and tear down VFs within the AS.

Messages are passed only between co-ordinators of adjacent ASes; no message is relayed across a co-ordinator, though information in a received message may contribute to a transmitted message. BGP, the Border Gateway Protocol, is a candidate for distributing Antidose signalling, as it follows the same interaction model. Antidose co-ordinators must also exploit information derived from existing BGP exchanges to determine whether specific Antidose signalling messages should be acted upon. If a neighbor AS reports that target T is requesting help with an attack, the receiving AS should ignore that report while it does not route traffic to T via the reporting AS; the report is presently deemed *inactionable*. This means that, insofar as BGP routing information is correct, an AS will never act upon a bogus Antidose signal that does not originate from T.

AS conformance to Antidose is set at five levels:

CL0 The AS does not interact with neighbors regarding Antidose.

CL1 The AS propagates only proof-of-work parameters, unconditionally to neighbors.

CL2 The AS propagates only proof-of-work parameters, selectively to neighboring sources of traffic to T.

CL3 The AS propagates proof-of-work parameters to all CL2+ neighbors, and FC parameters to all CL3+ neighbors (Figure 8).

CL4 The AS propagates all parameters as per CL3, and maintains VFs internally according to them.

A CL4 AS gathers all actionable reports for T (from gateways 3 and 4), and computes parameters for its internal VFs. It also computes parameters to be reported to upstream gateways (1, 2 and 5), those not used to deliver to T. For upstream

gateways contributing little traffic towards T, the AS has the option to propagate only proof-of-work parameters, with the result that FC parameters only propagate as far as ASes on the saturation boundary. When only propagating p/w parameters, they do not need to be propagated until some traffic for T is received. The sum of all b parameters reported upstream should not exceed the sum of b parameters from actionable reports; each c and k should be the latest available by serial number. As new reports come in, or become actionable or inactionable, or as the status of gateways changes, the AS re-gathers the reports and re-computes new parameters to distribute to its VFs and upstream neighbors. An AS has the option to defer propagation to upstream ASes to allow parameters from multiple neighbors to converge.

In addition to essential VF configuration parameters, an AS may propagate statistics concerning traffic to T upstream. These allow an upstream AS to determine whether traffic it is not filtering is contributing significantly to the attack. Statistics may also be propagated downstream, so that the target agent can determine that an attack is over. It may initially respond by increasing b and observing no detrimental effect, before fully tearing down the Antidose instantiation.

An AS does not need to apply VFs to all incoming traffic. An upstream gateway (e.g., 5) could be contributing little to an attack, so filtering on its traffic would be superfluous; additionally, clients using that gateway might not have been provided with p/w parameters, and so would have difficulty initially getting through the filter. However, if such contributions are aggregated with filtered traffic, a downstream AS might attempt to filter it anyway, and subject traffic of clients not participating in p/w to filtering they are unprepared for. To mitigate this, neighboring ASes could agree to instantiate a virtual gateway. The upstream AS would agree to send its filtered and unfiltered traffic through distinct virtual gateways, so the downstream AS could handle them separately. If the upstream AS has several unfiltered inputs, it could transport them internally over separate channels, and deliver them downstream each via a distinct virtual gateway. As a result, every distinct crossing of the saturation boundary could be represented in the most downstream AS as a distinct virtual gateway. As distinct gateways, a downstream AS could report different Antidose parameters and statistics through each one. This could be valuable when a DoS attack is not so distributed, as routes predominantly not used by attacking flows need not have the full Antidose mechanisms applied to them, including distribution of p/w parameters to clients.

CL1 and CL2 are intended for ASes with the highest performance and the least programmability, as forwarding p/w parameters is merely a signalling function. CL2 ASes have some basic monitoring capability to detect significant amounts of traffic to T; CL1s have none.

IV. Implementation

We have implemented the VF component[12] in a restricted C, and translated it to eBPF using the Clang compiler. eBPF is a virtual machine language developed out of Berkeley Packet Filters, chosen for BPFabric as an alternative to OpenFlow filters for its greater flexibility, its target independence, and its statically verifiable bounded execution time, making it suitable for high-speed switching-fabric execution.

The cookie set and proof set are each a pair of Bloom filters [21], with elements designated 'old' and 'current'. Each cookie or proof is checked against both elements of its corresponding pair of Bloom filters; if not present, the entity is added only to 'current'. Periodically, the 'old' set is discarded to be replaced with 'current', and 'current' is reset. Proofs and cookies are also rejected if their timestamps show them to be too old or too new. A proof or cookie can therefore only be accepted if it is both timely and not a duplicate. This strategy prevents replay attacks without having to record known entities indefinitely, which would result in saturation of a Bloom filter (or require unlimited memory for other techniques).

The use of Bloom filters obviates dynamic memory requirements. A Bloom filter with m bits, k hash functions and n entries has a false-positive probability $p \approx (1 - e^{-\frac{kn}{m}})^k$. An optimal k can be determined for given m and n as $k_{\text{opt}} \in \left\{\left\lfloor \frac{m}{n} \ln 2 \right\rfloor, \left\lceil \frac{m}{n} \ln 2 \right\rceil\right\}$. With each hash function yielding a w-bit index, we can address $m = 2^w$ slots in a Bloom filter. We choose $w = 16$ so $m = 65536$, requiring 64 kbit per Bloom filter, or 64 kB in total. With an anticipated $n = 10000$ unique cookies or proofs per refresh period, k_{opt} yields 5, so we generate five 16-bit hash indices for any given cookie or proof. This yields an FP rate of $p_{\text{opt}} \approx 4.3 \times 10^{-2}$ under such conditions.

The whitelist is a counting Bloom filter consisting of m 4-bit counters. The 'weight' of each unique valid cookie is used to increment selected counters (overflows are clamped to the maximum value) to represent increasing presence in the whitelist. Periodically, every counter is shifted right one bit, producing an exponential decay. Periodically issued cookies should produce linear growth, which then competes with the exponential decay to reach equilibrium. Except for false positives, this guarantees that a tuple in the whitelist will disappear within a fixed time from the point that the target agent chooses to stop refreshing it, and this time is related to the rate at which the agent re-issued cookies, not for how long it had issued them.

With $w = 17$, the whitelist's $m = 131072$ 4-bit counters occupy 64 kB. With $n = 10000$ currently whitelisted clients, $k_{\text{opt}} = 9$, so we generate nine 17-bit hash indices for any client-identifying tuple. This yields $p_{\text{opt}} \approx 1.8 \times 10^{-3}$, so less than 0.2 % of randomly addressed attack traffic should escape a VF under such conditions.

The hash function used for all Bloom filters is SHA-256, and each 256-bit hash is sliced into k w-bit fields to emulate several 'independent' hash functions, with the remaining $256 - kw$ discarded.

SHA-256 is also used for signature verification. The encryption suite is µECC[13], a small C implementation of ECDSA[14] chosen for its minimal dependence on other libraries. These

[12] Antidose source code is available at http://scc-forge.lancaster.ac.uk/svn-repos/seccrit-internal/antidose/.

[13] https://github.com/kmackay/micro-ecc

[14] Elliptic Curve Digital Signature Algorithm

are not compiled into eBPF, but are presented as external functions to the main VF implementation. This is a trivially feasible architecture for a software implementation of eBPF, and can be designed to exploit the hardware cryptographic functions of hardware network devices.

V. Evaluation

This section considers anticipated and measured behavior of Antidose under attack conditions.

A. Impact on DDoS strategy

Unaware of Antidose, a DDoS attacker has several behavioral options:

- Send packets directly to the target in an effort to saturate its inbound links, but using spoofed source addresses, so the target cannot identify the attacker.
- Send packets with the attacking node's real address, but without following usual protocol rules, e.g., any flow control, again to saturate inbound links.
- Send packets with the source spoofed as the target to reflectors, so that they issue unsolicited packets to the target and saturate links.
- Send packets adhering to the protocol, but attempt to engage the server in unproductive tasks, or make it use up its outgoing bandwidth supplying unwanted data, i.e., semantic/application-level attacks, not necessarily bandwidth-volumetric.

Antidose is intended to deal with the first three cases, because they are ordinarily difficult to deal with (a result of such behavior actually being consistent with good network utilization from the network operator's viewpoint). By default, Antidose blocks everything until clients demonstrate good behavior. In the first three cases, only packets from whitelisted addresses get through in volume, and none of which result in whitelisting of the used addresses. In the last case, the target can already reliably block an attacker by the identity it had to use to adhere to the protocol, and any whitelisting of the identity will expire without continuous consent of the target.

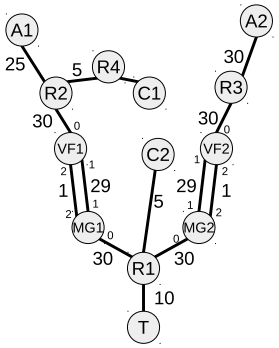

Fig. 9. Evaluation topology N1

To demonstrate the functionality of Antidose, we examine a network topology N1 modelling activity near a saturation boundary (Figure 9), and a broader topology N2 modelling the cumulative effect of a hierarchy of VFs on coalescing traffic (Figure 10). In topology N1, target T is accessed by two legitimate clients C1 and C2. Two nodes A1 and A2 flood UDP traffic to T in an attempt to saturate the link R1-T (the saturation boundary), and link capacities are set such that this occurs exactly when A1 and/or A2 are attacking. R* nodes are Ethernet learning switches. Each VF*-MG* structure models a single node on which a VF runs. In N2, T sits at the base of a 3-level hierarchy of VF*-MG* structures (each modelling the filtering efforts of a distinct AS) with a fan-out of 2. Link capacities are in Mbit/s.

A VF* node contains H and VF components (c.f. Figure 2). Port-0 traffic passes through H first, and then either goes to port 1 directly (if the source address does not match), or to the VF. PASS and PRIORITY traffic from VF go to port 1, and THROTTLE to port 2, which has considerably restricted capacity, and models the B component. All traffic entering on port 1 goes directly to port 0. No traffic is expected to enter on port 2. MG* nodes forward port-0 traffic to port 1, and all other traffic to port 0, so all PASS, PRIORITY and THROTTLE traffic from a VF is merged, and appears as a single port to any learning switch below it.

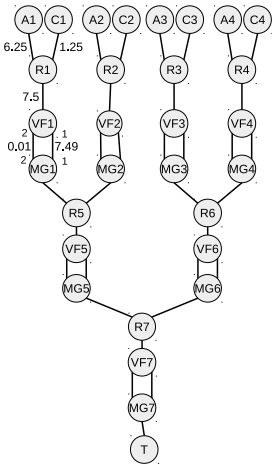

Fig. 10. Evaluation topology N2

The topologies are implemented in Mininet, using BPFabric softswitches for each switch node. The BPFabric controller is configured to load a simple merging eBPF function for MG* nodes, an antidose H+VF eBPF function for VF* nodes, and a simple learning-switch eBPF function for R* nodes. The controller interface allows the user to load map data to any given node to configure it, and commands are piped to it from the Mininet script. Learning switches build up their own internal configuration automatically, and mergers have no configuration. Antidose nodes must be provided with T's address and a public key to verify cookies.

Time (s)	Event
10	A1 starts attack
18	VF1 enabled
25	T cookie to C1
47	A2 starts attack
52	T cookie to C1
58	VF2 enabled
78	Stop

TABLE I
SCHEDULE FOR EXPERIMENT 1

In our first experiment, a Mininet script schedules the events shown in Table I on topology N1. TCP bandwidth is measured using `iperf` as a server at T, and as a client at C1/C2. `iperf` is used at A1/A2 to send UDP at a rate greater than those nodes' link rates to simulate a bandwidth-saturation attack. Cookies are generated using the program `antidose-mkcookie`.

The bandwidth measurements demonstrate the effectiveness of Antidose from two viewpoints. In Figure 11, we see that both C1 and C2 lose connectivity when A1 starts at 10 s. However, when VF1 is activated at 18 s, C2 recovers connectivity, and only loses it again at 47 s, when A2 starts. It regains it once more a little after 58 s when VF2 is activated.

C1 does not recover until about 28 s, having received and returned a flow cookie from T. On its return, it passes through VF1, resulting in C1's whitelisting at that point. Without refreshment, however, it loses it again at about 42 s when that cookie's effect on VF1's whitelist fully decays. (4-bit counters,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2018.2828938, IEEE Transactions on Network and Service Management

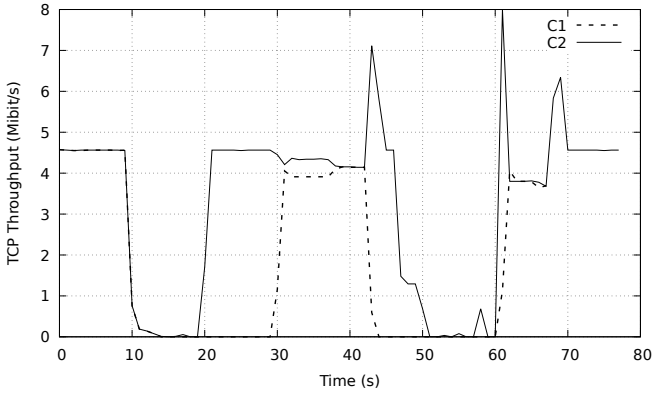

Fig. 11. TCP throughput in Experiment 1

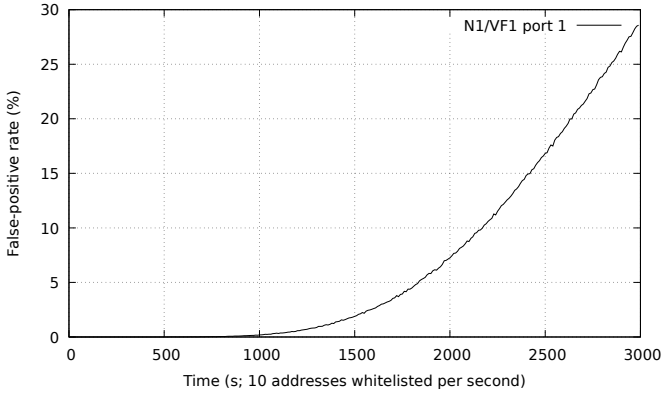

Fig. 12. False positives in Experiment 2

a half-life of 5 s, and a cookie weight $\geq 2^4 - 1 = 15$ yields a total decay time of 15–20 s.) The second cookie at 52 s enables a similar recovery (decaying at around 67 s)[15], but it is not effective until A2 traffic is blocked by VF2 from 58 s.

These traces demonstrate that, so long as filtering occurs before saturation, Antidose will allow target-identified clients outside the saturation boundary to break through filtering that would otherwise achieve the attacker's likely goal of removing all connectivity to T. Furthermore, T must continuously identify legitimate clients to prevent decay from whitelists, so T can effectively de-whitelist a previously well-behaving client with mere patience. Meanwhile, clients that do not have to go through filtering to reach T benefit from filtering on rival streams.

In the second experiment, we use `hping3` to send 120-byte SYN packets from A1 towards T in N1 as fast as it can. A filter is enabled at VF1 at time 0 s, and then a flow cookie *response* is generated every 0.1 s for a new IP address, and transmitted from C1 towards T[16]. The half-life of VF1 is set high so that no entry will decay before the end of the experiment. 30 000 distinct addresses are eventually whitelisted, and we count the number of TCP packets to T passing through VF1's larger link

[15] `iperf` stops reporting when it loses the connection, and does not begin again before the simulation finishes, hence the C1 trace finishes here.

[16] C1 emulates thousands of clients by spoofing their source addresses, avoiding performance limitations in Mininet.

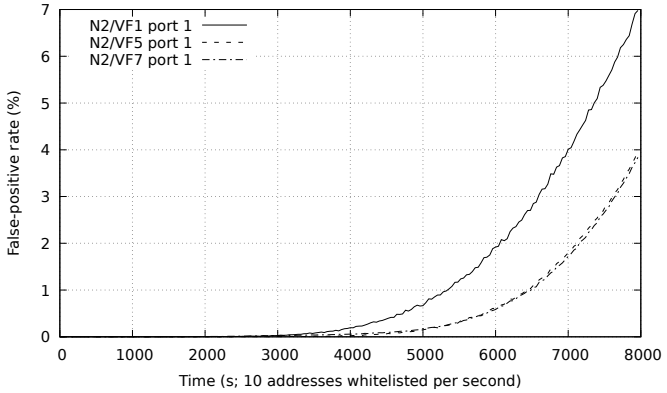

Fig. 13. False positives in Experiment 3

(which must be false positives) in 0.1 s bins, compute false-positive rates (FPRs), and plot them in Figure 12. As designed, after 10 000 clients at about 1000 s, FPR is still practically zero. At 3000 s, when 30 000 clients have been whitelisted, FPR is about 28.6 %, which roughly concurs with a computed $p \approx (1 - e^{-\frac{kn}{m}})^k \approx 29\,\%$, with $m = 131072$, $k = 9$ and $n = 30000$.

In our third experiment, we measure the impact of multiple layers of VFs on FPR, using topology N2. Total attack-traffic volume remains the same as in Experiment 2, but is split across four attackers A1–A4. Similarly, cookies are generated at 10 per second, but cycled across C1–C4. 80 000 addresses are eventually whitelisted, so that each VF at the top level receives 20 000, each at the middle level receives 40 000, and the sole VF7 at the bottom receives all. We capture attack traffic on the port 1s of VF1, VF5 and VF7, as shown in Figure 13, to measure FPRs of VF1 alone, the VF1-VF2-VF5 system, and the complete system respectively.

As VF1 experiences only the 20 000 whitelist entries from C1, its FPR after 8000 s is at most around 7%, and this is consistent with Figure 12 up to 2000 s (in which the same number of addresses where whitelisted in a quarter of the time). VF2–VF4 will behave similarly.

VF5's whitelist eventually contains 40 000 entries (from C1 and C2 combined), so its individual FPR should be around 55 %. However, the combined attack traffic it receives from A1 and A2 has already been reduced to 7%, so the *combined FPR of the VF1-VF2-VF5 system* is about $55\,\% \times 7\% \approx 3.8\%$, close to the measured value of 4 %. The VF3-VF4-VF6 system will behave similarly.

VF7's whitelist eventually contains 80 000 entries (from C1–C4), so its individual FPR is 96 %, so the combined FPR is $96\,\% \times 3.8\% \approx 3.7\%$, again close to the measured value, 3.8 %.

In summary, while VF7's whitelist is saturated beyond its design, the upstream VFs closer to the saturation boundary have already effectively reduced the attack volume.

Figure 14 shows performance measurements of the Antidose eBPF module running in the same DPDK environment with 10 Gbit/s ports as used in [15]. With Antidose in place, but no filter enabled within it, traffic not directed at the target almost reaches line rate with 128-byte packets. With a filter enabled

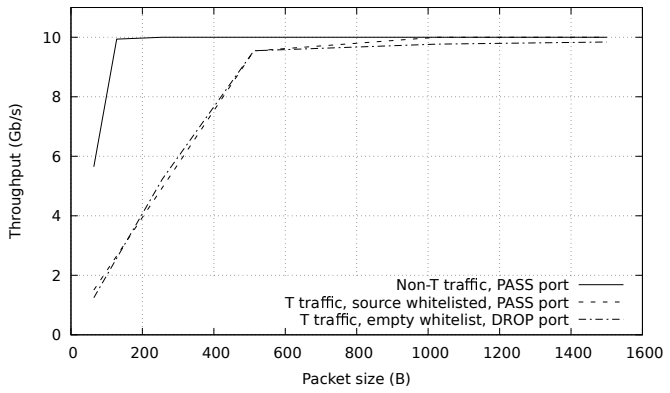

Fig. 14. Throughput in Experiment 4

and one IP address whitelisted, traffic from that source to the target almost reaches line rate with 512-byte packets. We also observed similar performance with unwhitelisted traffic to the target, regardless of entries in the whitelist up to 10 000.

B. Counter-attacks

An attacker that is aware of Antidose could attempt to by-pass or game its mechanisms.

An attacker can by-pass a VF by attaching proofs-of-work, but each must be unique, and only buys the successful delivery of one packet. The attacker can only deliver packets into the saturation zone at the rate it can generate proofs.

An attacker could attach proofs to the largest packets to maximize bandwidth consumption. However, large proof-carrying packets could be routinely dropped, as proofs are normally only useful for small connection-initiating packets.

Proofs are tied to source addresses, so attacking hosts cannot share proofs they have found with each other, unless they also spoof source addresses (and regardless of how many hosts share a proof, the number of packets that get through VFs on the saturation boundary will be at most the number of those VFs). The attacker might also have large amounts of processing power to generate proofs. However, if the target agent suspects either of these strategies, it can arbitrarily increase the difficulty of generating a proof without increasing the verification effort in VFs. Meanwhile, legitimate clients are also penalized, but two unique proofs could buy them membership of the whitelist, after which no more proofs are required. (Clearly, this counter-strategy is limited by how long W a new client is prepared to wait to form its first connection. With a packet size limit of $Z = 86\,\mathrm{B}$ (a SYN including Ethernet header and a modest p/w), and $W = 8\,\mathrm{s}$, an army of $N = \frac{BW}{2Z} \approx 12.5 \times 10^6$ nodes is required to sustain an attack of (say) $B = 2\,\mathrm{Gbit/s}$.)

An attacker can by-pass a VF by initially behaving well (resulting in whitelisting), and subsequently behaving badly by attempting to consume bandwidth within the saturation zone. If it does so with spoofed source addresses, it achieves nothing, as only packets using the whitelisted address as source get through. If it uses its unspoofed, whitelisted source address, but fails to adhere to protocol flow control, the traffic gets through and the bandwidth is consumed. However, such packets are more likely to reach the target or its agent, which may allow them to determine that the whitelisted address is being abused. They can then locally blacklist the address, withdraw refreshment of the address in whitelists, and await its decay.

The attacker could by-pass a VF by behaving correctly continuously, but attempt to engage the target in useless activity. This strategy is not covered by Antidose, and can be very successful because it is still difficult to determine if any given service request is genuine or bogus; an attack might be able to overcome any discriminator at this level simply by looking increasingly like a flash crowd. However, if a reliable discriminator is applied, the remedy is simple: terminate the connection, and block future connections from the host in the local firewall. The attacker cannot subvert this without returning to the lower-level attacks, which Antidose covers. The problem has therefore been reduced to devising an application-level discriminator (albeit still a very difficult problem) to be applied in the target server.

An Antidose-aware attacker has some additional options, attacking the remedial mechanism directly, including the following:

1) *A number of malicious clients behave normally, and get themselves whitelisted, with the aim of saturating whitelists to the extent that they produce more false positives.* The target agent would be unable to respond to traffic arriving on FP addresses, as it has not whitelisted them, so it cannot unwhitelist them.
   While this form of attack remains challenging, the attacker faces some difficulty with Antidose in place. Traffic must pass through several VFs from the saturation boundary to the target, and each may manage its whitelist differently, making it more difficult for a given source IP address to pass through all, especially as the attacker cannot predict which IP addresses will pass through any given VF as a false positive, and more so if the attacker is unable to spoof. With CL4 support expanded sufficiently beyond the saturation boundary expected under naïve attack conditions, this chain of VFs could be increased, and attack traffic could be prevented from coalescing sooner, so that the initial VFs are less saturated.
   Additionally, under this scenario, Antidose has achieved its goal of forcing an attacker to make its nodes' real addresses conspicuous to the target agent. If the target has a means to identify maliciously unproductive interactions, it can block their IPs locally, refuse new connections, refuse to refresh those IPs in whitelists, and await their exclusion from the attack.
2) *At least one attacking node transmits a flood of fake flow cookies, to force engagement of the cookie verification step in a VF.* The VF cannot resort to sampling flow cookies, as it will likely then miss many legitimate ones drowned out by fake ones. This option is available because no credentials are required to generate a fake cookie.
   If an attacker can command such a high volume of fake-cookie traffic at a given point, then it must be within

the saturation boundary, and so VFs should exist further upstream, where the fake-cookie flows have not yet aggregated to the same degree, and are still competing with regular traffic. Therefore, VFs should not have to cope with fake cookies at line rate, as a significant proportion of that rate would not be cookie traffic. Downstream VFs are protected from cookie floods because upstream VFs drop fake cookies.

Nevertheless, if fake-cookie floods are found to be burdensome, alternative cyphers with less demanding verification processes could be chosen, and need not be indefinitely unbreakable, so long as keys can be updated faster than they can be cracked.

3) *At least one attacking node transmits a flood of duplicate valid flow cookies, to force engagement of the cookie-uniqueness step in a VF.* This option is available because even a malicious client can obtain a valid cookie.

   This is a lighter-weight step than verification, and occurs before it, so if verification performance is sufficient, so is uniqueness. The first cookie will whitelist the client. Duplicates then pass the whitelist (but do not reinforce the client's presence in it), and are passed on with high priority, increasing the likelihood that they will be observed by the target agent, who can refuse to issue further cookies to the offending client. Lacking replacements for new cookies, the original will eventually expire, be regarded as broken, and be dropped.

4) *Attacking nodes obtain valid cookies, and share them with other attacking nodes, with the aim of saturating whitelists in VFs farthest from the target.* This option is available if attackers can spoof source addresses, otherwise they would not be able to transmit other clients' cookies. The attacker only has to find a number of key locations from which to transmit to ensure that the cookies are seen by every edge VF, so this approach does not require the attacker to use large amounts of bandwidth.

   As mutual duplicates following different but converging paths, the cookies will eventually be observed by some VFs more than once, and will be relayed under high priority. They are then more likely to be observed by the target agent, which can behave as under counter-attack 3, and refuse to refresh cookies to the hosts identified by the duplicates.

We note that in attacks 2, 3 and 4, attacking nodes are required to generate unusual packets that might make them more conspicuous to their local source networks. These packets are distinguishable without having to inspect them deeply or record a great deal of state.

For attacks 3 and 4, it might seem undesirable to give attacking packets (duplicate valid cookies) higher priority. However, under this priority, these packets are more likely to reach the target agent, to which they positively identify compromised (or even complicit) nodes in the attack, and which can then withhold new cookies.

The success of attacks 1 and 4 depend on the resistance of the whitelist to saturation, where a Bloom filter's false-positive rate becomes excessive. $m = 2^{17}$ 4-bit counters occupies 64 kB, and $k = 9$ hash functions and $n = 10000$ distinct entries yields $p \approx 1.8 \times 10^{-3}$ FP rate, which we believe is still effective at blocking unwhitelisted addresses during these more sophisticated attacks. Variations in the hash functions at different VFs in the chain from any given attacking node to the target will permit an even smaller proportion of attack traffic getting through. Furthermore, attacking nodes cannot predict which client identities will manifest false positives.

VI. Discussion

We discuss a number of open issues, and potential problems and solutions.

Inter-AS communication reliability during attack: It is a premise of Antidose that each AS has a reserved or priority channel with each of its immediate neighbors over which it can signal p/w and cookie parameters. Without such a channel, conveying $\langle c, k, b \rangle$ reliably over saturated links might be impossible. In our favor, the quantity to be transmitted is small and infrequent, and travels in the opposite direction to the attack flow. Also, as direct signalling with indirect neighbors is not required, the exact mechanism can be decided bilaterally, and an AS can ensure that only its authorized coordinators can transmit on a channel, and thus assure its peer of what it will receive.

VF deployment options in 'deep' networks: Two requirements compete in deploying a verification filter. First, it must be deployed sufficiently deep in the network to be beyond the saturation boundary. Second, the deeper the network, the less willing its operator is to deploy complex forwarding logic beyond examination of the destination host address, as capacities aggregate, and speed dominates requirements. Even with hardware data-plane programmability such as proposed by BPFabric, the overhead of a VF deployment might be too great for the network in which it must be deployed. However, the H component (Figure 2) has been consciously separated from other functionality to suggest a solution in such high-performance and low-programmability environments. H is fundamentally a destination-host forwarding function, so it already exists, even in deep networks. It can be used to separate attack traffic to auxiliary devices that have greater programmability, and so are capable of hosting VF. This is feasible so long as the attack volume exceeds the capacity of neither the channel that carries the separated traffic to VF, nor the receiving hardware itself. Extended forwarding functionality offered by the likes of OpenFlow and BPFabric could also help, e.g., source hashing for load-balancing across multiple VFs, and tunneling for reaching non-adjacent VFs.

Algorithmic alternatives: Bloom filters could be inadequate for some scenarios, e.g., where a large number of legitimate clients must be simultaneously whitelisted, yet false positives must still be minimized, and memory is tight. Antidose does not prescribe how a VF should implement its cookie sets, proof sets and whitelist, so other data structures (including eBPF tables) and algorithms can be unilaterally selected.

When very low FP rates are required, Bloom filters are not terribly efficient at storing sets of small keys, such as IP-protocol tuples, as is the case with the whitelist. To achieve

(say) $p = 0.0001$ with $m = 131072$ and $k = 9$, then $n \leq 6485$. Storing 6485 entries as a simple array would require only $(5 + 1) \times 6485 < 38\,\text{kB}$[17], which is somewhat less than the 128 kB required for a Bloom filter with 2^{17} 8-bit counters. However, data structures with better look-up/insertion performance are required, and correspondingly might have more comparable memory requirements. Hardware often present in routers, e.g., CAMs (content-addressable memory), could also act as whitelists.

Even with small keys, a Bloom filter still has some favorable characteristics:

- It has fixed size, so no dynamic memory allocation is required.
- Its complexity (excluding the hash computation) is $O(k)$, whether checking or inserting, and $O(m)$ for decay. (The number of inserted entries n has no impact on performance.)
- It can cope with larger and variable key sizes (e.g., IPv6 addresses) trivially (without any extra memory or complexity).

Other asymmetric cyphers may be used for cookie signatures, with verification performance being a key factor. The inter-AS protocol must be able to indicate the cypher type as well as the public key. Excessive variety should be avoided, however, as all CL4 ASes must be using the same cypher (or a cookie would have to contain multiple signatures).

SHA-256 could be expensive for generating Bloom filter indices, especially as its cryptographic quality is redundant. Non-cryptographic algorithms such as SipHash [22] are usually lighter-weight (potentially improving throughput of checking smaller packets against whitelists), and hash combination techniques can be used to generate a series of indices with no significant loss of accuracy [23]. A cryptographic hash is still required for p/w computations and cookie signatures, but there is also no benefit to sharing partial SHA-256 hash states between Bloom-filter index generation and proof/signature verification, unless the amount to be hashed is at least 64 bytes, as the internal hash state only changes on every 64-byte block.

Good behavior of non-TCP protocols: Antidose is designed with the TCP handshake in mind as an indicator of good behavior. However, it leaves detection of a successful handshake up to the target agent, so it is not TCP-specific, and any protocol with a similar handshake phase should be compatible. Some legitimate interactions that do not involve handshakes also can be dealt with. For example, under a DNS reflection attack, most of the errant DNS replies will not be let through VF nodes, as they are from hosts not legitimately communicating with the target. However, should the target need to directly perform an external DNS look-up of its own, it will be initiating the communication to external DNS servers, and can pre-emptively whitelist them before sending the DNS request. (Note also that no proofs of work are required for this, and a slightly more sophisticated H (e.g., selecting all UDP *:53 to T:* traffic) could allow non-DNS traffic to bypass VF.)

Lack of client conformance: It is unlikely that sufficient momentum would be gathered to upgrade a substantial number of clients with both necessary behaviors of injecting proofs and responding to flow cookies (although the latter is already standard behavior, just one that is often disabled out of security concerns). However, it is the client agent that must actually perform these tasks and it does not need to be precisely co-located with the client, only close enough to see all of its traffic and modify it. The client's ISP is in an ideal position to do this, and can act for multiple clients, so the problem is reduced to a per-ISP upgrade, rather than per-client. Indeed, this is the argument behind BCP38 in RFC2827 [24], whereby an ISP filters its customers' outgoing traffic to eliminate its contribution to source-address spoofing.

Unlike BCP38, in which one ISP's actions directly benefit only other users, Antidose support in ISPs does not require large-scale participation to benefit a participating ISP's customers. In general, each individual client benefits from participating unilaterally, regardless of whether any other clients also participate. Furthermore, participation is not required at all under normal circumstances.

ICMP Echo exposure/NAT: ICMP Echo is chosen as a pre-existing means to accurately deliver flow cookies to only the whitelists on the path from subject to target, even in the face of route changes and asymmetric routing. However, hosts may be unwilling to respond to pings as a security risk, though it should be trivial to distinguish response-worthy cookie pings if the host initiated the interaction with the ping sender. In cases where the ping sender is initiating new interactions (e.g., where the target is a DNS client), the receiver (a DNS server) is already exposing itself to provide its service, and is no more exposed by responding to echo requests with similarly-sized echo replies. More widespread client participation would also open up the possibility of a dedicated protocol as an alternative to ICMP Echo. Hosts behind Network Address Translation (NAT) share a single public address visible to the target. Consequently, it is the NAT's public address that is whitelisted and all hosts behind it could be denied access due to bad behavior of one. This can only encourage users behind shared addresses to ensure their systems are free of malware.

CL4 conformance requirement: The highest level of conformance is required of all ASes within the saturation zone. The zone is determined by the transmission capability of the attacker, and the capacity of the ASes nearest to the target. Antidose has been designed to impose minimal processing burden on ASes so that deployment remains favorable even as one attempts to push it deeper into the network, while also ensuring that it does not open a new attack vector on such sensitive devices. Whether Antidose is deployed in practice depends on network operators' willingness to deploy emerging technologies such as BPFabric, and on whether the corresponding costs of this deployment are outweighed by the costs of other extensions of infrastructure laid out in anticipation of future DDoS attacks.

Saturation boundary transparency: If the source of the attack is not well distributed, some ingresses to the saturation zone might not require filtering, so clients using these ingresses will not need to be whitelisted, nor to attach proofs. The b parameter is intended to permit a small amount of such unfiltered traffic, but it will be ineffective as filtered

[17] 4 bytes for the IPv4 address, 1 for the protocol, and 1 for the counter

and unfiltered flows merge downstream. Potentially, an AS could avoid this merging by relaying unfiltered flows through alternative gateways, which need not be physically distinct. The downstream AS can then issue upstream different b values to distinct 'virtual gateways' according to their contribution to an attack, allowing more for those that appear to be contributing minimally. Each AS also could have the option to avoid merging unfiltered flows, and present them as distinct virtual gateways downstream. This would allow the 'shape' of the saturation boundary to be transparent to the target AS, as each entry point to the saturation zone would arrive through a distinct virtual gateway to the target AS, which could then configure each one with distinct values of b.

Proof-parameter propagation: Antidose depends on legitimate clients receiving proof-of-work parameters in order to reliably make initial contact with the target server, and this requires widespread CL1+ conformance, at least between the agents of the target and each client. As the set of clients of any given host is not generally known a priori, this necessitates global CL1+ conformance, and consequently global flooding of p/w parameters! This is unlikely to happen, so are there any alternative parameter-distribution mechanisms that can be relied upon under an attack? If parameters are not to be propagated AS-by-AS, receivers must be able to verify that a challenge is genuine, as an attacker could attempt to pollute potential clients with fake challenges. Also, the mechanism must itself not be prone to (D)DoS (which potentially rules out DNS).

First, we can defer propagation of p/w parameters until a need arises. A client's initial packet's arrival into an AS could trigger the propagation into the supplying neighbor AS, though this then requires that the client try to send more packets to complete the propagation. This still requires global CL1+ conformance, so a bridge is needed to cross the gulf between the AS and the client agent directly. An AS that knows that the next AS is CL0 could issue parameters in a packet directly addressed to the client. To prevent abuse, it would have to keep track of which clients have been recently informed (to avoid becoming part of a reflection attack), and include details of the triggering packet (so that the recipient could use it with confidence).

VII. Related work

Many DDoS remedies have been proposed over previous decades. Under SIFF (Stateless Internet Flow Filter) [25], intervening routers drop packets to the target if they do not carry a 'capability', a sequence of numbers which, when indexed by the packet's TTL, yields a code arbitrarily chosen by the router. A legitimate client forms a correct sequence (one matching the codes of routers the packet will pass through) by issuing an explorer packet, to which intervening routers add their codes. The target sends the explorer packet with a complete capability back to the client, so only a client that did not spoof can receive the capability. This scheme faces a 'denial-of-capability' problem, in that explorer packets might not reach the target because they compete for bandwidth with attack packets. Explorer packets cannot be given higher priority, as they attacker could then use them itself.

Portcullis [26] applies the concept of 'proof of work' (a means by which clients may demonstrate legitimacy to a target without its prior knowledge of them) to connection initiation. When applied to (say) SIFF, clients add proofs to their explorer packets, and which then automatically receive higher priority. Attackers could do the same, but as unique proofs are required, they can only attack at the rate that they can generate proofs.

In dFence [27], middleboxes are deployed away from the target to separate traffic belonging to legitimate connections from potential attack traffic. Each middlebox pretends to be the target, and responds directly to incoming connections on its behalf. If a connection is established, it pretends to be the client, relaying the connection to the target. Further, it protects itself from SYN attacks by employing SYN cookies. VFence [28] proposes a very similar system, but exploiting Network Function Virtualization (NFV) as a means to dynamically instantiate middleboxes.

'Flow cookies' are introduced in [29] to allow a target to permit traffic from a specific client to pass through a middlebox. Cookies are generated by the target, and cannot easily be faked. They are issued to clients, who attach them to their packets for middleboxes to receive and verify. Packets not containing verified cookies are disfavored by the middlebox.

In [30], inter-domain collaboration is proposed to block identified attack flows through commands propagated on reverse paths towards a source. It identifies the risks of coarse filtering leading to loss of legitimate traffic, and the need for confident inter-domain mitigation signalling. FlowSpec [31] specifies a means for one domain to identify flows by both source and destination that require special handling in another domain. The vision of the IETF Working Group DOTS [32] includes domains under attack appealing to upstream domains to report DDoS problems, expecting help in their mitigation, and receiving status reports to indicate when to withdraw this help. Inter-domain colaboration exploiting increased programmability for the purpose of improved security continues to be explored [33], [34].

Antidose combines several of the above concepts. A participating router keeps a whitelist, and populates it on receiving flow cookies issued on behalf of the target. Proof-of-work is used on initial connection packets from clients not yet whitelisted. Proof-of-work and cookie parameters are passed between adjacent ASes, and trusted only while they apply to downstream gateways currently used to deliver traffic to the target.

VIII. Conclusion

We have presented Antidose, a scheme allowing participating ASes to mitigate the effects of a Distributed Denial-of-Service attack on a target, and which is able to control whitelists within ASes upstream of the saturation zone of the attack. Effectively, through interaction with only immediate neighbors, an AS with only a low-level network view of traffic is given the ability to discriminate legitimate packets from likely attack packets using criteria set by the target, which has a higher-level (transport or application) view. We have presented an implementation of Antidose's critical component,

the verification filter (VF), and analyzed its behavior in the face of various counter-attacks.

The Antidose VF is sufficiently computationally simple to be deployed in BPFabric, a restricted execution environment for switching fabric, with the heavy-weight operations of hashing and signature verification handled externally and therefore potentially in hardware. We demonstrated that, even in this restricted environment, the VF correctly discriminates traffic according to the target's ever-developing definition of legitimate and malicious peers, and that Bloom filters are effective as whitelists even when there are thousands of simultaneous or recent legitimate clients.

The environmental restrictions of BPFabric make it suitable for hardware acceleration (e.g., with NetFPGA), demonstrating the feasibility of deployment of Antidose in ASes with high-performance and low-programmability equipment. The techniques and principles employed by Antidose reduce the barriers to AS operators managing the automatic mitigation of bandwidth-saturating DDoS attacks. Practical and robust proof-delivery/whitelisting mechanisms remain open issues.

ACKNOWLEDGEMENTS

This work is carried out under the EPSRC-funded projects SAII (grant ref. EP/L026015/1), TOUCAN (EP/L020009/1) and MaaS (EP/N033957/1). The authors wish to thank Nic Hart for input on evaluation.

REFERENCES

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[2] M. Jonker, A. Sperotto, R. van Rijswijk, R. Sadre, and A. Pras, "Measuring the Adoption of DDoS Protection Services," in *Proceedings of the 2016 ACM Internet Measurement Conference, IMC 2016*. ACM, Nov. 2016, pp. 279–285.

[3] S. Sharwood, "GitHub wobbles under DDOS attack," http://www.theregister.co.uk/2015/08/26/github_wobbles_under_ddos_attack/, Aug. 2015.

[4] S. Khandelwal, "602 Gbps! This May Have Been the Largest DDoS Attack in History," https://thehackernews.com/2016/01/biggest-ddos-attack.html, Jan. 2016.

[5] M. Karami, Y. Park, and D. McCoy, "Stress testing the booters: understanding and undermining the business of ddos services," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 1033–1043.

[6] B. Schneier, "Lessons from the Dyn DDoS attack," https://www.schneier.com/blog/archives/2016/11/lessons_from_th_5.html, Nov. 2016.

[7] M. McKeay, "Q2 2017 state of the internet security report," Akamai, Tech. Rep., 2017. [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q2-2017-state-of-the-internet-security-report.pdf

[8] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet background radiation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 27–40.

[9] R. Beverly and S. Bauer, "The Spoofer project: Inferring the extent of source address filtering on the Internet," in *Proceedings of USENIX SRUTI workshop*, 2005.

[10] W. Scott, "POSTER: A Secure, Practical & Safe Packet Spoofing Service," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 926–928.

[11] S. Simpson, A. Lindsay, and D. Hutchison, "Identifying Legitimate Clients under Distributed Denial-of-Service Attacks," in *4th International Conference on Network and System Security*. IEEE, Sep. 2010, pp. 365–370.

[12] A. Goodney, S. Narayan, V. Bhandwalkar, and Y. H. Cho, "Pattern based packet filtering using NetFPGA in DETER infrastructure," in *1st Asia NetFPGA developers workshop. Daejeon, Korea*, 2010.

[13] F. Engelmann, T. Lukaseder, B. Erb, R. van der Heijden, and F. Kargl, "Dynamic packet-filtering in high-speed networks using NetFPGAs," in *Future Generation Communication Technology, 2014 Third International Conference on*. IEEE, 2014, pp. 55–59.

[14] A. Ghani and P. Nikander, "Secure in-packet Bloom filter forwarding on the NetFPGA," in *European NetFPGA Developers Workshop*, 2010.

[15] S. Jouet and D. P. Pezaros, "BPFabric: Data Plane Programmability for Software Defined Networks," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, March 2017. [Online]. Available: http://eprints.gla.ac.uk/138952/

[16] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computing Surveys*, vol. 39, no. 1, p. 3, 2007.

[17] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115–139, 2006.

[18] J. Niccolai, "Analyst Puts Hacker Damage at $1.2 Billion and Rising," https://www.computerworld.com.au/article/91948/analyst_puts_hacker_damage_us_1_2b_rising/, Feb. 2000.

[19] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[20] A. I. Ali, "Comparison and Evaluation of Digital Signature Schemes Employed in NDN Network," *International Journal of Embedded systems and Applications*, vol. 5, no. 2, Jun. 2015.

[21] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[22] J.-P. Aumasson and D. J. Bernstein, *SipHash: A Fast Short-Input PRF*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 489–508.

[23] P. C. Dillinger and P. Manolios, *Bloom Filters in Probabilistic Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 367–381.

[24] P. Ferguson, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827, May 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2827.txt

[25] A. Yaar, A. Perrig, and D. X. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2004, pp. 130–.

[26] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting connection setup from denial-of-capability attacks," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 289–300, Aug. 2007.

[27] A. Mahimkar, J. Dange, V. Shmatikov, H. M. Vin, and Y. Zhang, "dFence: Transparent Network-based Denial of Service Mitigation," in *Proceedings, 4th Symposium on Networked Systems Design and Implementation, Cambridge, Massachusetts, USA*. USENIX, 2007.

[28] A. H. Jakaria, W. Yang, B. Rashidi, C. Fung, and M. A. Rahman, "VFence: A Defense against Distributed Denial of Service Attacks Using Network Function Virtualization," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 02, June 2016, pp. 431–436. [Online]. Available: doi.ieeecomputersociety.org/10.1109/COMPSAC.2016.219

[29] M. Casado, P. Cao, A. Akella, and N. Provos, "Flow-Cookies: Using Bandwidth Amplification to Defend Against DDoS Flooding Attacks," in *Proceedings, 14th International Workshop on Quality of Service (IWQoS 2006)*, 2006, pp. 286–287.

[30] K. Argyraki and D. R. Cheriton, "Scalable Network-Layer Defense Against Internet Bandwidth-Flooding Attacks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1284–1297, 2009.

[31] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and D. McPherson, "Dissemination of Flow Specification Rules," RFC 5575, Aug. 2009. [Online]. Available: https://rfc-editor.org/rfc/rfc5575.txt

[32] R. Danyliw and T. Gondrom, "DDoS Open Threat Signaling," https://datatracker.ietf.org/doc/charter-ietf-dots/, Jun. 2015.

[33] D. Migault, M. A. Simplicio, B. M. Barros, M. Pourzandi, T. R. M. Almeida, E. R. Andrade, and T. C. M. B. Carvalho, "A Framework for Enabling Security Services Collaboration Across Multiple Domains," in *37th International Conference on Distributed Computing Systems*. IEEE, Jun. 2017.

[34] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and Elastic DDoS Defense," in *24th USENIX Security Symposium (USENIX Security 15)*. Wash., D.C.: USENIX Association, 2015, pp. 817–832.