SENSS Against Volumetric DDoS Attacks

Sivaramakrishnan Ramanathan
Jelena Mirkovic
USC/ISI
Los Angeles, CA
satyaman,mirkovic@isi.edu

Minlan Yu
Harvard University
Cambridge, MA
minlanyu@seas.harvard.edu

Ying Zhang
Facebook, Inc.
Menlo Park, CA
zhangying@fb.com

ABSTRACT

Volumetric distributed denial-of-service (DDoS) attacks can bring any network to a halt. Because of their distributed nature and high volume, the victim often cannot handle these attacks alone and needs help from upstream ISPs. Today's Internet has no automated mechanism for victims to ask ISPs for help in attack handling and ISPs themselves do not offer such services. We propose SENSS, a security service for collaborative mitigation of volumetric DDoS attacks. SENSS enables the victim of an attack to request attack monitoring and filtering on demand, and to pay for the services rendered. Requests can be sent both to the immediate and to remote ISPs, in an automated and secure manner, and can be authenticated by these ISPs, without having prior trust with the victim. Simple and generic SENSS APIs enable victims to build custom detection and mitigation approaches against a variety of DDoS attacks. SENSS is deployable with today's infrastructure, and it has strong economic incentives both for ISPs and for the attack victims. It is also very effective in sparse deployment, offering full protection to direct customers of early adopters, and considerable protection to remote victims when deployed strategically. Deployment on the largest 1% of ISPs protects not just direct customers of these ISPs, but everyone on the Internet, from 90% of volumetric DDoS attacks.

CCS CONCEPTS

• **Networks** → **Denial-of-service attacks**;

KEYWORDS

DDoS defense, IP spoofing, traffic filtering, collaborative defense

ACM Reference Format:
Sivaramakrishnan Ramanathan, Jelena Mirkovic, Minlan Yu, and Ying Zhang. 2018. SENSS Against Volumetric DDoS Attacks. In *2018 Annual Computer Security Applications Conference (ACSAC '18), December 3–7, 2018, San Juan, PR, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3274694.3274717

ACSAC '18, December 3–7, 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6569-7/18/12…$15.00
https://doi.org/10.1145/3274694.3274717

1 INTRODUCTION

Volumetric distributed denial-of-service (DDoS) attacks can overwhelm even the largest networks. In 2016, Dyn, a large and geographically distributed DNS management network, was hit by 1.2 Tbps attack [35], from more than 100,000 sources. In 2018, Github was hit by a 1.35 Tbps attack, from thousands of autonomous systems and tens of thousands of hosts [29]. Distributed attacks create problems not just for their victims, but also for the ISPs and bystanders, which share the path with attack traffic [6].

Volumetric DDoS attacks cannot be handled by the victim alone, and require help of other networks. When the attack's volume is so high that it overwhelms the victim's link to its ISP, or even a link within the ISP, this ISP or other upstream ISPs must help in the attack's mitigation. Similarly when the attack is indistinguishable from legitimate traffic, filtering at the victim has a high collateral damage. Upstream networks can help identify locations close to attack sources, and place filters at these locations, where they inflict a lower collateral damage.

Today's Internet has no *automated* mechanism for victims to ask their peers or remote networks for help in attack handling, and has low incentives for ISPs to offer such services. While most ISPs will help when engaged, asking for help occurs through human channels today, and the ISP's response is often limited to blackholing all traffic to the victim [32].

We propose SENSS, a framework for collaborative defense against volumetric DDoS attacks. Using SENSS, the victim of an attack can request help from direct and remote networks (called "ISPs" in the rest of the paper) in an automated and secure manner, and pay on demand for the services rendered. The victim sends messages to SENSS-enabled ISPs ("SENSS ISPs" for short), asking for traffic and/or route monitoring or control actions. Each message is signed by the victim and authenticated by SENSS ISPs, which prevents misbehavior and misuse. Replies to monitoring messages bring information to the victim, which it can combine with its own knowledge of its traffic and business priorities, to devise custom attack mitigation. The victim then issues control messages to SENSS servers to mitigate the attack. SENSS APIs at ISPs can be readily implemented by today's infrastructure, which makes SENSS cheap for ISPs to deploy and incentivizes deployment.

Our **first contribution** is the novel design of the SENSS framework, for secure, automated, on-demand collaboration between victims and ISPs (Section 3). SENSS has two main design principles:

(1) *Intelligence at the victim, simple functionalities at ISPs.* Many collaborative defenses place intelligence (e.g., deep packet inspection – DPI) in the cloud or at an ISP, which makes defense functionalities complex and costly. In SENSS, the victim drives all the decisions, such as what to monitor and

which actions to take to mitigate attacks. This allows for simple functionalities at ISPs, which are easily implemented in the current ISP infrastructure. Intelligence at the victim makes SENSS cheap for ISPs, and effective in sparse deployment. It also limits the impact of misbehaving participants (Section 3.5). Victims that lack technical skills to make decisions about attacks can delegate SENSS control to their ISPs or to third parties via SENSS's proxy mechanism (Section 3.5).

(2) *Versatile, evolvable and customizable defense.* Many defenses today work against one or a few attack variants. SENSS's simple APIs can be used as building blocks by the victims to create customized defenses against many DDoS flavors. As attacks evolve, these simple APIs can be used to build new defenses *on top of the existing SENSS infrastructure*.

Our **second contribution** lies in novel algorithms for handling of direct floods, reflector attacks and cross-fire attacks, all described in Section 3. These algorithms and SENSS enable the following novel functionalities that are not available in today's defenses: (1) mitigation of direct floods close to attackers, (2) surgical mitigation of reflector attacks, (3) mitigation of cross-fire attacks.

Our **third contribution** is a thorough evaluation of SENSS using simulation on the Internet's AS-level topology and emulation on the Deterlab testbed [2] (Section 4). SENSS is very effective in sparse deployment, offering full protection to direct customers of early adopters, and considerable protection to remote victims when deployed strategically. In case of 2016 attack on Dyn, SENSS deployment at Cogent, Level 3, Zayo and Comcast would have filtered 100% of the attack. In general case, deployment on the largest 1% of ISPs would protect not just direct customers of these ISPs but everyone on the Internet from 90% of direct floods and reflector attacks. SENSS further filters attacks closer to their sources, and saves twice as much bandwidth as cloud-based defenses. SENSS overhead within an ISP remains constant irrespective of attack's complexity or its volume, and SENSS's handling of client requests requires just a fraction of a second.

2 BACKGROUND AND RELATED WORK

In this section we discuss types of DDoS attacks that we would like to address, and the need for a collaborative defense between victims and ISPs. We also survey related work in the operational and research realms, and position SENSS in this landscape.

2.1 Volumetric Attacks

DDoS attacks overwhelm the victim with excessive traffic, sent from many sources. In this work we focus on three variants of volumetric attacks, which usually cannot be handled at the victim: direct floods, reflector attacks and cross-fire attacks. We use the term *floods* to refer to all three of these variants. Direct floods send traffic directly to the victim. Reflector attacks send request traffic to public servers (e.g., DNS servers), spoofing the victim's address, and the servers reply to the victim, overwhelming it. Cross-fire attacks send traffic to many destinations. This traffic congests some bottleneck link in an ISP, which is also shared by the victim's inbound traffic. In all three cases, there is a tree-like pattern of traffic, with attack and legitimate sources being the leaves and the bottleneck link being the root. This is illustrated in the Figure 1(a).

Because volumetric attacks only need to generate high enough volume to DoS the victim, they do not have to have a specific signature. They can also be spoofed. Further, they can be launched from many distributed sources. The 2016 Dyn attack [35] and 2018 Github attack [29] were launched from 10,000 – 100,000 of sources, distributed over thousands of networks.

2.2 Related Work

DDoS defense is a mature field. In this Section we focus only on defenses, which address volumetric attacks, and on collaborative DDoS defenses.

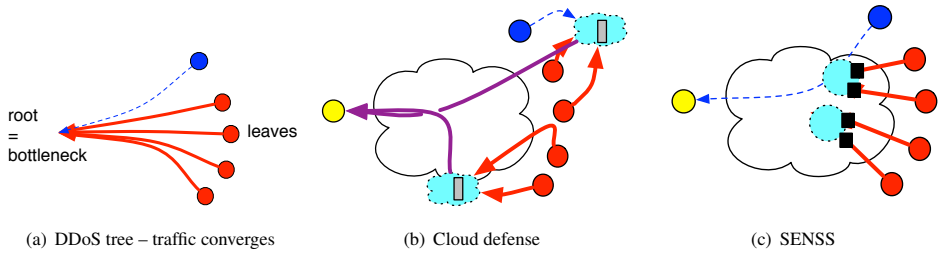
Victim-end defenses, such as Bro [27] or Arbor APS [25] are often used to detect and filter smaller DDoS attacks. However, large volumetric attacks cannot be handled by the victim, because the bottleneck is upstream from the victim's network.

First-ISP. The victim can engage its ISP through human channels, to help in mitigation, i.e., to filter the attack. Most ISPs offer only crude filtering of all traffic to the victim, aka *remotely-triggered black hole (RTBH)* [32]. This protects ISP infrastructure, but cuts the victim off the Internet and exacerbates the impact of the attack.

Bohatei [8] is a research defense within a single ISP. It uses software-defined networking (SDN) and network function virtualization (NFV) to offer flexible, elastic DDoS defense. It instantiates a number of VMs on demand, and steers victim's inbound and outbound traffic through these VMs, which implement custom mitigation programs. Bohatei's custom programs offer finer-grain traffic analysis at a higher resource cost (statistics storage and application header access) than SENSS. SENSS is complementary to Bohatei, as it handles spoofed and cross-fire attacks, which cannot be handled by Bohatei.

Cloud-based defenses, such as CloudFlare [6] and Zenedge [36], defend against attacks through geo-replication of the victim's resources. To effectively defeat floods, this geo-replication needs to be extensive, requiring equipment and peering contracts at many locations and a lot of excess bandwidth to withstand high-volume attacks. When a cloud wants to grow, it has to enter into more peering agreements and buy and install more equipment and fiber; both are expensive. Clouds *attract all the victim's inbound traffic to themselves*, and apply "packet scrubbing" to identify an attack signature and filter the attack traffic. Clean traffic is then tunneled to the victim. The scrubbing process is proprietary, but it often involves deep-packet inspection (DPI), and thus has a high processing cost.

Compared to clouds, SENSS enables *ISPs on the attack path* to offer a distributed attack response, through victim-ISP collaboration. ISPs deploy SENSS on *their existing infrastructure*, leveraging existing functionalities like SDN, Flowspec [22], Netflow and access control lists (ACLs). This makes SENSS's deployment costs smaller than that of clouds. On the other hand, SENSS filters are coarser-grain (network- and transport-level but not application- or payload-level) than those of clouds, which may lead to a higher collateral damage for application-level attacks. For this reason, SENSS focuses only on handling of volumetric attacks, which usually randomize fields after the transport header.



(a) DDoS tree – traffic converges (b) Cloud defense (c) SENSS

Figure 1: DDoS attacks and ways to handle them. (a) Attack traffic mostly converges at one bottleneck, forming a tree-like pattern. (b) filter in a cloud, requires redirection and dedicated infrastructure (c) filter on the path of the attack, with ISP's help, using SENSS

SENSS infrastructure grows when more networks adopt SENSS and start supporting it on its existing infrastructure. This makes SENSS growth more sustainable than cloud growth.

We illustrate handling of network floods by clouds and by SENSS in Figures 1(b) and 1(c) respectively. While clouds divert the victim's inbound traffic to their infrastructure, SENSS empowers ISPs on the traffic's path to help in attack diagnosis and mitigation. This saves more bandwidth (see Section 4.4).

Collaborative defenses. The need for collaborative defense against evolving DDoS attacks was elaborated by Kang et al. in [14]. There are several collaborative research approaches to DDoS mitigation, such as Pushback [12], traceback [10], StopIt [20], AITF [1], DefCOM [26], CoDef [19], SIBRA [4], SCION [28], TVA [34], Defense by Offense [33] and SPIFFY [15]. Some of these approaches are not deployable today, because they require Internet redesign – TVA and SCION. Other approaches are less deployable than SENSS, because they require router hardware changes. Traceback, StopIt, AITF, CoDef and DefCOM require packet marking at routers, and StopIt requires cryptographic operations on traffic.

Further, some approaches apply victim-to-source collaboration model – StopIt, Defense by offense and CoDef – which has the victim ask the source networks (legitimate or attack) for help in attack mitigation. Victim-to-source collaboration cannot be effective in sparse deployment, because many sources would reside in legacy networks. SENSS, on the other hand, is very effective in sparse deployment, thanks to its victim-to-ISP collaboration model. Even a few ISPs on the attack's path, collaborating with the victim, enable that victim to filter much of the attack traffic.

Pushback [12] applies *ISP-to-ISP* collaboration. Such collaboration is difficult in general, because an ISP handles a lot of traffic for many customers. Moderate floods, which may not be noticeable at the ISP level may overwhelm a customer on a small downlink. To detect these floods the ISP would have to continuously monitor all its customers' traffic, which is costly. It would also have to make guesses as to which traffic to filter, while a victim can make a better decision (and implement it with SENSS) based on its knowledge of its inbound traffic patterns.

CoDef [19] focuses on handling of cross-fire attacks, through collaborative rerouting and rate limiting. Collaboration occurs between the affected network and the networks that host legitimate sources. CoDef's mitigation (rerouting) resembles SENSS's route control messages, but its diagnosis requires packet marking, while SENSS does not require this. SENSS is thus more deployable than CoDef. Further, CoDef's collaboration model (victim-to-source) means that CoDef requires a wider deployment than SENSS to achieve a given effectiveness target.

SIBRA [4] introduces a new mechanism for legitimate sources and destinations to reserve bandwidth on inter-AS links, and thus isolate themselves from DDoS attacks. SIBRA is complementary to SENSS and attempts to prevent DDoS, while SENSS attempts to mitigate it. SIBRA also requires more extensive changes to ISPs than SENSS and is thus less deployable.

SPIFFY [15] introduces implicit collaboration between the bottleneck and sources of traffic, by temporarily expanding the bottleneck link and identifying sources, which do not expand their sending rate, as attackers. SPIFFY only handles cross-fire attacks and is thus complementary to SENSS.

2.3 SENSS vs First-ISP vs Clouds

We now briefly discuss how SENSS could complement current operational solutions: first-ISP and clouds.

SENSS's messaging mechanism offers a way for remote networks to collaborate on demand, without prior trust. As such, SENSS mechanisms could be used to remotely trigger first-ISP or cloud-based solutions. This could transform cloud defenses from pre-arranged into on-demand solutions.

SENSS could further be used when first-ISP or cloud defenses fail or are overwhelmed (e.g. Spamhaus attacks in 2013 [31] and Dyn attacks in 2016 [35]). A cloud or an ISP can use SENSS to achieve a collaborative response with other ISPs in the Internet, to lighten its load. SENSS can also benefit from clouds and first-ISP solutions, which could act as victim proxies, and enable adoption of SENSS among non-technical victims.

3 SENSS

In this section, we discuss challenges of collaborative defense, provide a high-level overview of SENSS operation, and detail how SENSS would be implemented at ISPs and at attack victims.

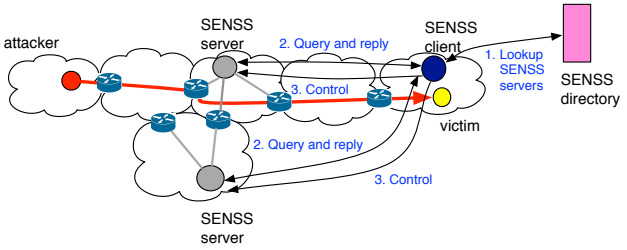


Figure 2: SENSS architecture.

3.1 Challenges

When designing a collaborative defense, we must address the labor-division challenge (what will be done where), the deployment challenge (how to motivate deployment) and the security challenge.

Labor division: we must decide which functionalities are required for attack handling and where to implement them. SENSS uses *victim-to-ISP* collaboration, which places all the intelligence at the victim and requires only simple functionalities of ISPs. This enables a victim to request help from any ISP for diagnosis and mitigation, but remain fully in control over the attack handling. The victim can combine its internal knowledge of its business and customers, with observations received from SENSS, to create customized defenses. Because the victim can directly contact any SENSS ISP, neighbor or remote, SENSS is very effective in sparse deployment (see Section 4) and robust to misbehavior (see Section 3.5).

Deployment: collaborative defenses must offer significant benefits to networks that deploy them. SENSS brings clear benefits to the victims of attacks, but the challenge lies in making it appealing to ISPs. We address this challenge in several ways. First, SENSS *handles several attack variants*, thus an ISP can offer it to current customers as added value, and many customers may find it useful. Second, SENSS *works with existing hardware* and thus has low cost to deploy. Third, SENSS *offers significant benefits to early adopters*, thus ISPs can offer immediate protection to their customers against DDoS, even in sparse deployment.

Security: collaboration should not introduce new vulnerabilities even under direct attacks, or when some collaborators misbehave. SENSS has multiple *security layers* to protect against misuse: (1) It allows the victims to only observe/control traffic and routes for their prefixes, whose ownership is proved via digital certificates. SENSS ISPs validate these certificates before processing each request. (2) All communication between the victim and the SENSS ISPs is secured using TLS, and is thus protected against message sniffing, forgery, modification or replay, (3) SENSS operation is driven by the victim, with each SENSS ISP reporting its own observations of the victim's traffic and routes. Such design severely limits the impact of a misbehaving ISP (see Section 3.5).

3.2 SENSS Architecture

Figure 2 illustrates SENSS operation. SENSS consists of client code, which runs at an end network or an ISP, and server code running at an ISP. Clients are illustrated as blue circles and servers as grey circles in the Figure. We refer to the client-deploying network as the *victim*, but it may decide to engage SENSS even in absence of attacks, e.g., to learn about its normal traffic patterns. An ISP can decide to provide SENSS services for free or for a fee, which could be a flat-rate or per-message fee. Economically, a per-message fee makes the most sense as the ISP incurs cost for running SENSS only when it handles SENSS messages, and a victim is only interested in paying for SENSS when they are under attack.

An ISP deploys SENSS by implementing SENSS APIs on a server, and exposing them to the public. These APIs can be implemented as a Web service, and thus deployed at the ISP's Web server. This leverages existing approaches for service robustness (Web server replication), message security (HTTPS), and charging for service (e-commerce). The ISP also implements automated mechanisms within its network, which act on SENSS client's requests by implementing traffic/route handling in border routers. This is shown as grey lines in the Figure. These mechanisms can be implemented as a collection of scripts, which communicate with switches/routers using Netflow, Openflow, Flowspec or switch-specific management language (e.g., for ACL setup). We assume that all client-server communication occurs at the network level and not at the host or user level. This limits the cost of communication and the state at SENSS servers.

A victim under attack runs the SENSS client. The client first uses a *public SENSS directory* to identify multiple SENSS servers (step 1 in the Figure). One way to implement this directory is to assign a common DNS name to each SENSS server. The victim sends *queries* to SENSS servers about the victim's inbound traffic or the routes to the victim's prefixes (step 2 in the Figure). For security reasons, clients are only allowed to ask about and manipulate traffic and routes for the prefixes their network owns. While some attack diagnostics may be easier if we allowed clients to ask about anyone's traffic, this would create grave privacy concerns, and jeopardize adoption of SENSS.

Each query is accompanied by a digital certificate, which proves the that the client is authorized to issue query and control messages about the IP prefixes contained in the query. We call this certificate *proof of prefix ownership* and provide more information about it in Section 3.5. The SENSS server validates the certificate and the message, performs the required service, charges the victim for it, and returns the response to the client (step 2 in the Figure). The server may also decide not to perform a given action, e.g., because it would be against some internal policy or because it would consume too many resources. In this case the server does not charge the victim and simply returns a *reject* message to the client.

The client analyzes responses obtained from SENSS servers, identifies the best action and the best locations for mitigation (e.g., filtering, rerouting) and issues *control* messages to chosen SENSS servers (step 3 in the Figure), which authenticate them, charge for and perform the actions. The query and control steps can be repeated multiple times, until the desired mitigation effect is achieved.

3.3 ISP Implementation

APIs. Table 1 summarizes SENSS APIs, used by the client to request SENSS services. Each message has an *action* field, which specifies if the traffic/route handling rules should be installed (*start*) or removed (*stop*) at the SENSS ISP. The server enacts these handling rules after each message, by installing them in the appropriate border

Type	Message	Matching Fields	Action	Reply/Response
Traffic query	*traffic_query*	*predicates, otime*	*start/stop*	*rule_id* and a list of <tag, direction, #bytes/#pkts> matching the *predicate* during *otime*
Route query	*route_query*	*prefix*	none	*rule_id* and AS paths from the SENSS ISP to the *prefix*
Traffic control	*filter/allow*	*predicates, tag, dur*	*start/stop*	*rule_id* and filter/allow all traffic matching the <*predicate, tag*> for duration *dur*
Route control	*demote*	*prefix, seg, dur*	*start/stop*	*rule_id* and reduce pref. of routes to *prefix* if they contain *seg* in AS path, for duration *dur*
Abort	*abort*	*prefix*	none	delete all rules for the prefix, and blacklist the public key

Table 1: SENSS APIs

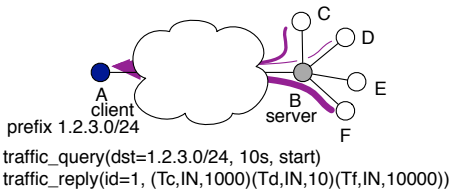


Figure 3: Illustration of traffic query.

routers at the ISP. The server then replies with a unique identifier (*rule_id*) to the client. All rules expire after the duration (*otime* or *dur* parameter) specified in the message. When a rule expires at the server, the server removes the rule from the routers where it was installed. The client may choose to remove a handling rule prior to its expiry, by setting the *action* field to *stop* and specifying rule's identifier. Additionally, the client can use a special *abort* message, which removes all the rules for the given prefix. A SENSS server may also decide not to handle the client's message, and return a *reject* reply to the client.

The basic building block of traffic messages (*traffic_query* and *traffic_control*) are *predicates*, which match a flow based on header fields. For example, the predicate "(src_ip=10.0.0.1 | src_ip=10.0.0.2) & src_port=53" matches flows with source IP 10.0.0.1 or 10.0.0.2 and source port 53. Predicates support conjuction (&), disjunction (|), negation (!) and wildcard (*) operators. Predicates can also specify traffic direction (SELF – generated by the ISP, IN and OUT), and they can specify a *tag*, denoting a peering link to which the query applies. A *traffic_query* asks for monitoring of traffic matching the predicate for time *otime*. On receiving a *traffic_query*, a SENSS server installs the appropriate traffic collection rules (Openflow or Netfilter) to some or all border routers. After time *otime* the server collects observations from the routers and removes the collection rules. It then aggregates these observations into a single reply, and sends it to the client. The reply contains *rule_id*, and a list of tuples. Each tuple contains a *tag*, identifying the neighbor, traffic direction *dir* and the amount of traffic observed in packets or bytes. The tag is the ISP-specific, unique identifier per neighbor, and should be anonymized in a manner, which is reversible only by the ISP. For example, the ISP could encrypt the neighbor's identity with a secret key. If the client later requests filtering of traffic with a given tag, the ISP decrypts the tag to identify the appropriate peering link and the switch, where the filters should be installed. One traffic_query and its reply are illustrated in Fig. 3. The client A asks for monitoring traffic to its prefix, 1.2.3.0/24, for 10 seconds. The server B replies with the list containing three records – one for each neighbor that forwards traffic to B with destination IP in 1.2.3.0/24.

A *traffic_control* message requests a SENSS ISP to *filter* or *allow* traffic matching the specific predicates and tags, for the duration *dur*. If the client sends multiple messages for the same prefix, they will be matched in the order in which they are received. The server translates each rule into a filtering rule that its routers understand (e.g., ACL, Flowspec, Openflow) and installs it at the routers matching the *tag* field from the message. Rules are removed after time *dur*.

A *route_query* asks an ISP about the AS paths in its best routes to the prefix. The server collects the best routes from all border routers, and replies with the set of unique AS paths.

A *route_control* message asks an ISP to demote all its routes for the select prefix that contain the AS-path segment *seg*, for the duration *dur*. The server collects all the routes from each border routers for the specified prefix. It then decides for each router if the route should be demoted. This decision can take into account the ISP's internal route selection policy and the cost of demotion (e.g., if a customer route is being demoted and the next best route is peer or provider route there will be an associated cost). If the server decides to demote the route, it issues appropriate messages to the router's BGP daemon. After time *dur* these messages are reversed.

Identifying deployment routers. After receiving and validating (see Section 3.5) a SENSS message, the SENSS server identifies the routers or switches, which should implement the required functionality. For traffic queries with IN direction, and for route queries and route control, all ingress routers would be used. For traffic filters, the server only needs to identify those ingress routers that were specified in the *tag* field of the client's message. If no tag is specified, the egress router to the specific prefix can be used for filter installation.

Implementing monitoring and control at routers. If the ISP supports SDN, SENSS can implement all traffic observation and control functionalities by issuing OpenFlow messages to the SDN controller. For ISPs that do not support SDN, a SENSS server can enact traffic observation by installing Netflow collection rules in routers. For cost reasons, the server can install rules that use packet sampling, and then extrapolate the actual packet/byte rates for the traffic_reply. Finally, each router supports access control lists (ACLs). SENSS server can enact traffic monitoring of a flow by installing an ACL rule with "permit" target, and later querying the number of packets that matched the rule.

The server can implement traffic control at non-SDN ISPs by using ACLs or Flowspec [22]. The SENSS server implements route observation and route control by using BGP software. For route observation, it queries the given router for its best route to the select prefix. For route demotion, it first queries the given router for all its routes to the select prefix. Then it decides which, if any, routes to demote and issues commands that lower the values of the routes' local preference attributes.

Practical issues. SENSS messages lead to rules being installed at a router's TCAM, whose space is usually limited. We find that many distributed attacks can be handled efficiently with very few rules per router, using coarse flow specifications on fields such as neighbor tag, destination IPs, transport ports, protocols, etc. On the other hand, the client may rarely want to observe and control flows at a fine-grained level, using the source IP field. Since the number of rules, which use source IP field, could quickly skyrocket, ISPs could

discourage use of such rules by pricing them much higher than other rules. In Section 4 we show that DDoS attacks can be handled well with coarse-grained rules.

An ISP may have privacy concerns about others learning about their traffic handling and routes. We have carefully designed SENSS to avoid leakage of information that ISPs consider private or sensitive, such as traffic load balancing and peering.

(1) *Route replies reveal only public peering information,* which is already visible in BGP advertisements.
(2) *Traffic replies are anonymized or aggregated.* An ISP can return encrypted tags in traffic replies that ask only about IN direction, as we illustrate in our examples for flood w/o signature in Section 3.4. When the victim sends filter requests, the ISP decrypts the tags to identify where to place filters. In traffic queries that ask about IN and OUT traffic (e.g., cross-fire example) the ISP can return aggregated traffic ("all" tag in our example for cross-fire) or omit traffic volume. This protects confidential load balancing information.

An ISP may not be open to others changing its routing decisions in any way. Such an ISP can always return the *reject* response for *demote* messages. This will make it unable to aid in cross-fire attack mitigation, but it will still be able to help in handling direct floods and reflector attacks.

3.4 Client Programs

We now illustrate how a victim could design custom attack mitigation programs using SENSS APIs, by presenting *four* programs for mitigation of the following attacks: direct floods with and without transport/network signature (we use the terms *flood w sig* and *flood w/o sig*), reflection and cross-fire attacks. All are novel contributions, made possible by the SENSS APIs (Table 2).

Detection of DDoS attacks and identification of the attack signature is out of scope of our research. The victim can use existing tools, such as AMON [13], Packetscore [17], Bro [27] or Arbor APS [25].

Floods w sig. If the SENSS client has a network/transport signature (e.g., the target destination IP and port number) to separate legitimate from attack traffic, it uses SENSS to send *traffic_filter* messages containing the signature and the target prefix to SENSS servers. Figure 4(a) illustrates handling of floods with signature. To preserve budget, the client can install filters only on some servers that are likely to carry most of the attack, e.g., Tier 1 and Tier 2 ISPs, and install them one by one, until mitigation is achieved.

Floods w/o sig. When DDoS attack traffic is very similar to the legitimate traffic or very diverse or spoofed, the victim cannot derive a sufficiently specific TCP/IP header signature, to separate the attack from the legitimate traffic. Using SENSS the client can observe the differences in the geographical distribution of the victim's inbound traffic prior and during the attack. Such differences usually exist, because both the legitimate and the attack traffic's distribution over source networks are heavy-tailed. This can be used for *location-based filtering*, where the client identifies ISP-ISP links (tags) where filter placement would minimize collateral damage. During normal operation, the client periodically issues *traffic_queries*, and extracts the tags and the volume of traffic for the ISP-ISP links – pre_i. Let pre_T be the total inbound traffic prior to the attack. During attack, *traffic_queries* are repeated, to learn the tags and volume of links that now carry a mix of attack and legitimate traffic – $post_i$. Comparing pre_i and $post_i$, our program identifies candidates for filter placement (i) as those tags where $post_i$ is large but pre_i was small or zero. The program uses a parameter $\alpha \in (0, 1)$ to crudely control anticipated collateral damage. It orders tags by their pre_i values, in an increasing order, and selects first N so that $pre_{i1} + pre_{i2} + ... + pre_{iN} \leq \alpha \cdot pre_T$.

Figure 4(b) illustrates handling of floods without signature, with tag names starting with letter "t" (anonymized) and reported traffic volume shown in parentheses. We highlight the tags that are candidates for filtering, assuming $\alpha = 0.2$. This example also illustrates how the client can patch together a partial view of the DDoS tree even in sparse SENSS deployment.

Reflector attacks. Reflector attacks are challenging to handle, because the victim wants to receive replies to its legitimate service requests, and must filter replies to spoofed traffic, but these two kinds of replies are very similar at the network level (e.g., in Figure 4(c) the victim wants to receive the blue traffic and filter the grey).

We handle reflector attacks by the victim network "marking" all its query traffic, by NATing all its legitimate requests for targeted service S, using a small range of port numbers R=[a, b]. NATing can be done at the border router of the victim's network. NATing creates an artificial TCP/IP signature, which persist in replies. We can use this signature to filter out replies caused by attack traffic, because these replies will not go to the NAT IP address nor to the port numbers in the range [a,b].

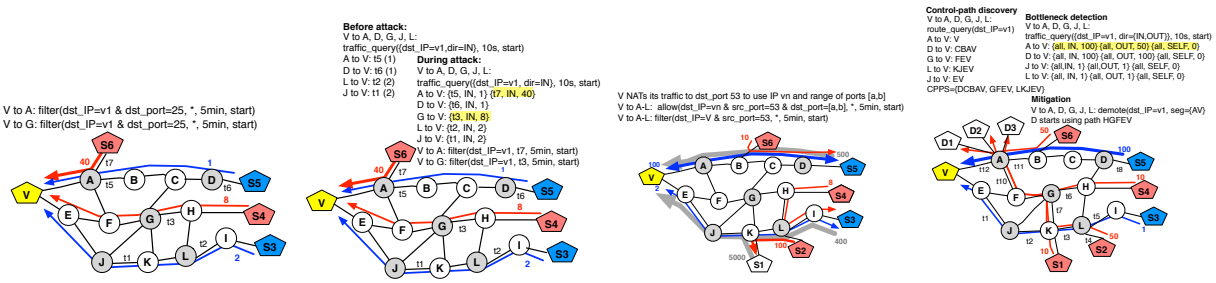
The SENSS client installs two filters at each selected SENSS server, which will be applied in this order: (1) to allow all traffic to the NAT on ports in range [a,b], from service port for S, (2) to deny all other traffic to victim's prefixes coming from service port for S. This surgically removes all of the attack and has no collateral damage. Range [a,b] must be chosen so that it is small enough to make guessing hard, and large enough to accommodate regular request traffic. The client further must frequently change the range [a,b] to avoid guessing and DNS hijacking attacks. We analyzed traffic from public traces (MAWI [11]) and found that 99.9% of networks could keep around 300 ports in the range [a,b], and change them each 10 seconds to satisfy these conditions. Figure 4(c) illustrates handling of reflector attacks.

Cross-fire. The cross-fire attack [16] creates congestion at an ISP upstream from the victim. For example, in the Figure 4(d), attack networks S1, S2, S4 and S6 send traffic to D1, D2 and D3, and the congestion occurs at A, causing V's inbound traffic drops.

SENSS client can: (1) identify AS path segments that host the bottleneck and (2) mitigate the attack by re-routing around the bottleneck. To *identify* the bottleneck AS path segment the client first issues *route_queries* to select SENSS servers, and extracts the AS paths from the replies, forming the control plane path set – CPPS. The client then traverses each path in the set issuing a *traffic_query* for SENSS server in each AS (if such server exists) on the path. If a SENSS ISP has multiple ASes, the server replies separately for each AS specified in the query. The client compares the outgoing traffic from the upstream reports with the incoming traffic from the downstream reports. It identifies the bottleneck segment as the segment where the upstream AS reports sending more traffic than was received by the downstream AS. The client *mitigates* the attack by issuing demote messages containing the bottleneck link to select SENSS servers. To be robust against lying servers, the client includes

Attacks	Key Idea	SENSS features we use
Flood w sig	Filter traffic based on signature	filter
Flood w/o sig	Identify geographical differences between usual and attack traffic paths; use for filter placement	traffic_query before and during attack, filter
Reflector DDoS	Victim marks traffic; ISPs filter non-marked traffic	filter, allow
Cross-fire	Locate bottleneck links via traffic queries and route around them. Combines route/traffic info.	route_query, traffic_query, demote

Table 2: Key innovations of client programs and the SENSS features we use



(a) Flood w sig (b) Flood w/o sig (c) Reflector (d) Cross-fire

Figure 4: Illustration of DDoS attack handling with SENSS. Yellow nodes are victims, blue are legitimate clients and red are attackers. White nodes are ISPs that do not deploy SENSS, and grey are ISPs that deploy SENSS. Red and blue lines represent attack and legitimate traffic, respectively. Grey lines represent replies to spoofed requests. Numbers in the same color above the lines represent volume in Gbps. Black tags on links are ISP-specific, anonymized tags that will be used in some replies by SENSS. We show only relevant fields in SENSS messages, shown as text above the diagram.

the reporting AS in the bottleneck segment. Figure 4(d) illustrates handling of cross-fire attacks. We have highlighted the identified bottleneck, which will be demoted. Query replies have aggregated traffic in reports to protect ISP's load balancing decisions.

3.5 Security and Robustness

In this Section we detail how we have secured SENSS against direct attacks and misuse.

Securing communication: SENSS only allows the victim to issue messages about its own prefixes. The *proof of prefix ownership* is publicly verifiable, unforgeable information, which binds the SENSS client's public key with the list of its prefixes. The client includes its prefix-ownership certificate with each request. Upon successful certificate verification, the ISPs cache the {public key, prefixes owned} information for some short time.

We can create proof of prefix ownership by using RPKI ROA (Route Origin Authorization) certificates. SENSS ISPs deploy RPKI certificate verification. Using RPKI certificates enables SENSS servers to verify requests from remote clients, with whom they have no prior trust. While a client would still have to establish a payment mechanism to pay for services, this can be automated using e-commerce solutions.

RPKI is today deployed in a limited fashion, mostly because networks see no special benefit of RPKI in sparse deployment. If SENSS used RPKI to very prefix ownership, this may provide added incentive for deployment. Today's increasing deployment of MANRS [21] also motivates larger deployment of RPKI. As an alternative to RPKI certificates, which contain much additional information not needed by SENSS, we could design new certificates, which bind a public key with prefixes owned. These certificated would be issued by the same entities that today issue RPKI ROA certificates, i.e., authorities that have assigned a given address space to the victim.

The communications between a SENSS client and a SENSS server are secured using TLS [7], and occur via HTTPS. The victim uses the public key from the proof of prefix ownership in the TLS key exchange process.

If the private key corresponding to the proof of prefix ownership gets compromised, the attacker could control all of the victim's inbound traffic. To reduce the risk of this, the SENSS client would issue the *abort* message to all SENSS servers as soon as it becomes aware of the compromise. This message, when successfully authenticated, purges all traffic/route rules for the given prefix at a given SENSS server, and the server removes all corresponding rules from the routers. The SENSS server also blacklists the public key, which it used to authenticate the message. This stops the use of the stolen key. The SENSS client can access SENSS servers again when it acquires a new certificate. The reverse scenario, where the attacker issues the abort message using a stolen private key, gives no advantage to attacker. It cuts the attacker off SENSS, while the client can still access it using its new certificate.

Robustness and SENSS proxy: During attacks, the victim's network may become unable to receive replies from SENSS servers. The victim can outsource its decision making power, along with its customized mitigation programs, to a *SENSS proxy* — a machine in a different ISP, e.g., a public cloud, that will act as SENSS client. The victim may set up one or several proxies, prior to any attack, as a backup service. The victim generates a prefix-ownership certificate for the proxy using its private key, binding the proxy's public key with one or more of the victim's prefixes.

The proxy monitors the victim's service availability, e.g. by periodically issuing requests for some public service offered by the victim. When the availability declines, the proxy starts attack mitigation. If the victim has information about the attack (e.g., type, TCP/IP header signature) it can communicate it to proxy using one-way messages, e.g., using DOTS [23] or custom UDP. These messages carry a unique ID to avoid replay, and are encrypted by a key shared between the victim and the proxy. The proxy waits for the signature for some limited time. If the signature is received, the proxy activates the custom program for handling flood with signature. Otherwise, it activates the the custom program for handling flood without signature. The proxy includes both its and the victim's original certificate in the SENSS messages it sends. The SENSS server validates the certificates by validating the entire certificate chain.

Some victims may not be sufficiently technically savvy to detect attacks or make mitigation decisions. These victims can offload their attack mitigation to their first-hop ISP or to a cloud-based DDoS defense, by creating a SENSS proxy there.

An attacker could target a SENSS server to disable its operation. An ISP can replicate SENSS server functionality for robustness, just as it is done today with other services.

Handling Misbehavior:

SENSS clients have low incentive to misbehave. Clients are unlikely to send excessive requests to the server because they need to pay for each request. SENSS security mechanisms further ensure that client actions only affect traffic and routes to their own prefixes. However, a SENSS ISP cannot check if a victim is under attack, and it does not need to. Any network can use SENSS to require its traffic and routes to be handled in certain way by upstream ISPs, even when it is not under attack. It is up to ISPs to set the pricing scheme in such a way to fully recoup their costs of running SENSS, and to discourage excessive messages.

SENSS servers could lie about their observations or fail to implement control actions, for which they have charged the victim. In replies to *traffic_queries* a SENSS server may claim that it sends or receives more or less traffic than it does, and it may report a fake distribution of traffic over real or fake tags. In replies to route_queries a SENSS ISP may make up AS path segments. Table 3 lists all the possible ways a server may lie in its reports (column 2), and the wrong decisions the victim may make (column 3).

The overall effect that a lying server has on SENSS operation falls into only two categories: Legacy or Dropper. A Legacy liar has no effect on victim's traffic, but the victim pays for its reports. For example, if a SENSS server lies about its traffic to make it smaller, its effect is that of a legacy ISP. Legacy liars prolong the attack mitigation and make it more costly for the victim, but they cannot make the victim drop legitimate traffic, postpone attack mitigation indefinitely or influence victim's actions at other ISPs. A Dropper liar can drop some victim's traffic due to its lying, e.g., through the victim installing unnecessary filters *at the lying ISP*. For example, if a SENSS server reports a higher traffic on its links, it may create a dropper effect. Dropper liars are already on the data path and can drop traffic even without SENSS, so SENSS does not make the situation any worse.

Attack	Message (Lie)	Action, effect at liar
Flood w and w/o sig	TR (IN < Actual)	No filter, Legacy
	TR (IN > Actual)	Filter, Dropper
Reflector	None	
Cross-fire	TR (OUT = IN)	None, Legacy
Cross-fire	TR (OUT > Actual)	Demote larger seg., Legacy
	RR (fake seg.)	Demote fake seg., Legacy
	TR (fake seg.)	Demote fake seq., Legacy

Table 3: Lying ISP scenarios and their effect (TR: traffic query reply, RR: route query reply)

A server may fail to render the requested services, but still charge the victim for them, thus increasing own profits. A server could also extort the victim by dropping its traffic and then charging for diagnosis and mitigation. Both of these attacks are made possible by SENSS, because it allows the ISPs to charge victims when handling SENSS requests. While we cannot prevent these attacks, we sketch here how a victim can build a reputation score for each server, and use it to avoid underperforming or extortionist servers. The victim can monitor the effect of each control message by measuring the traffic it receives after the message was accepted and processed by a SENSS server. Control messages, which fail to reduce attack traffic would indicate underperforming servers. The victim can use each instance of underperformance to internally assign some bad reputation points to the given SENSS server. After a while these would accrue, allowing the victim to identify and avoid such servers in the future. Similarly, the victim can detect Dropper ISPs by running our client program for cross-fire handling. If a given AS is a part of the bottleneck segment, the victim can assign some bad reputation points to this AS. When the reputation declines significantly, the victim can conclude that that AS is a Dropper ISP and the victim can use SENSS to demote routes containing this AS.

4 EVALUATION

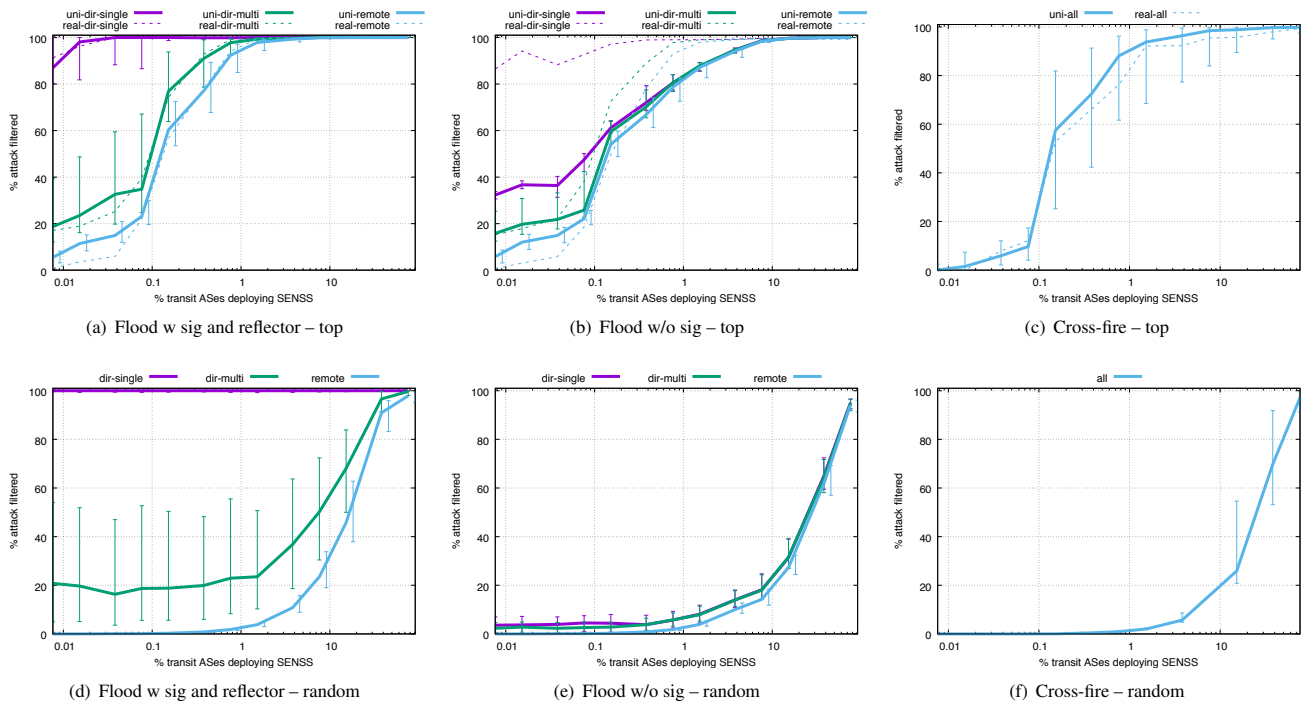
In this Section we first evaluate SENSS's effectiveness in sparse deployment, using a *simulation*. We use an AS-level simulator of the Internet, developed in Perl. We first illustrate how SENSS would help in an attack scenario similar to the 2016 attack on Dyn [35]. We then evaluate SENSS's effectiveness in sparse deployment and show that: (1) direct customers of SENSS ISPs have immediate benefits and are fully protected against direct floods and reflector attacks – this is an excellent deployment incentive for ISPs, (2) strategic deployment at 0.1–0.8% of all ISPs (0.3–3% of transit ISPs) protects everyone against 90% of floods, and (3) SENSS outperforms cloud-based DDoS defenses, saving 2–4 times more bandwidth.

We next evaluate SENSS's response speed, scalability and overhead, using a SENSS *prototype* on the DeterLab testbed [2], in the 186-switch, Cogent topology from TopologyZoo. SENSS's message processing delay scales linearly with the number of switches and concurrent requests and it is 0.05–0.25 seconds per request.

In this section we use AS and ISP terms interchangeably, for simplicity.

4.1 Evaluation Methodology

Simulation methodology. For our effectiveness tests, we infer AS-level topology and routing from CAIDA's [5] AS relationships



(a) Flood w sig and reflector – top (b) Flood w/o sig – top (c) Cross-fire – top

(d) Flood w sig and reflector – random (e) Flood w/o sig – random (f) Cross-fire – random

Figure 5: DDoS attack filtered, versus the percentage of transit ASes deploying SENSS, given the top and the random deployment strategy.

dataset, from May 1st, 2017. There are 57,552 ASes, 114,018 customer-provider links and 133,795 peer-to-peer links. Routing is inferred using the no-valley, customer-prefer approach [9].

In each attack instance we deploy SENSS on some ASes, following one chosen deployment strategy, and we simulate legitimate and attack traffic flows as aggregates on inter-AS links. We simulate just the aggregate rate of each flow and not its packets, nor details (e.g., source and destination addresses, ports or transport protocol). When we simulate flood w sig, we assume that attack traffic is so different from the legitimate traffic, that it is possible to devise a header-level signature for its filtering. Installing a filter on some SENSS ISP in the simulation of flood w sig only removes attack traffic going to the victim. When we simulate flood w/o sig, we assume that attack and legitimate traffic are so similar that no header level signature is possible. Installing a filter on some SENSS ISP in the simulation of flood w/o sig removes both legitimate and attack traffic going to the victim. We measure the amount of legitimate and attack traffic dropped and the bandwidth consumed by the attack on inter-AS links. We perform 1,000 random trials for each data point and show the median (lines) and 25th and 75th percentile (errorbars).

Emulation methodology. For our evaluation of response speed, scalability and overhead we have developed a SENSS prototype, including client and server functionalities, and the client customization programs described in this paper. We deployed our prototype on the DeterLab testbed [2] and measured the time it takes to serve a SENSS request under many concurrent requests. We replicated the Cogent topology (186 nodes) from the Topology Zoo[18]. On each node we ran Quagga as router software and Open vSwitch as the SDN software. We used RYU as the SDN controller. There was one SENSS server in the topology. For control plane requests, it connected to the Quagga software on switches via telnet, and for data plane requests it sent OpenFlow messages. We emulated large, 100 ms, end to end propagation delays between the SENSS server and each victim.

On the Cogent topology we use gravity model [30] to emulate legitimate and attack traffic. We generate legitimate traffic using iPerf (TCP mode) and attack traffic using a custom tool to generate UDP flood. We generate sufficient legitimate traffic to fill a 1 Gbit link from our victim to the ISP and then launch the attack.

4.2 2016 attack on Dyn

We reproduce the attack on Dyn in 2016 by reproducing locations of Mirai bots, using bot IP addresses from [24]. We divide 1.2 Tbps equally among bots, then allocate each bot to its AS, based on the bot's IP address. We use 7015 and 13977 as victim ASes for Dyn. We then strategically deploy SENSS on only four ASes, which are close to Dyn and on path between most Mirai bots and Dyn. These are AS 174 (Cogent), 3356 (Level 3), 6461 (Zayo Bandwidth) and 7922 (Comcast). With that deployment, and if attack signature were possible, SENSS would filter 100% of the attack traffic with only four filtering rules (one per ISP). Cogent filters 56% of the attack, and Zayo filters 40%, so even two SENSS ISPs would filter almost all the attack.

We also consider the case when attack signature is not possible. We simulate legitimate traffic to Dyn by assuming that it flows mostly from large US connectivity providers. We extract the top 10 providers from a 2017 survey [3], and simulate legitimate traffic proportionally to each provider's number of customers. Using α=5%, SENSS filters 99% of traffic to AS 13977, and 90% of traffic to AS 7015. This requires around a 1,000 filtering rules, with 60% being at Cogent and 40% being at Level 3, one rule per POP.

4.3 Effectiveness in Sparse Deployment

We now investigate how deployment strategy and the number of deployment points influence effectiveness. We investigate two deployment strategies. In *top strategy*, we deploy SENSS at the top $N = (1...10,000)$ ASes, ordered in decreasing order by their customer cone size. In *random strategy*, we deploy SENSS at the random $N = (1...10,000)$ ASes. In both cases, we only consider deployment at ASes that have at least one customer link, i.e. transit ASes. There are 13,123 such ASes in our topology – 23% of all ASes.

Uniform traffic: Since we cannot anticipate where the attackers, legitimate clients and the victim may reside, we deploy them at random. We first randomly select a victim, and then randomly select 1,000 ASes to host attackers and the additional 1,000 ASes to host legitimate clients. We distribute attack/legitimate traffic equally among attackers/legitimate clients.

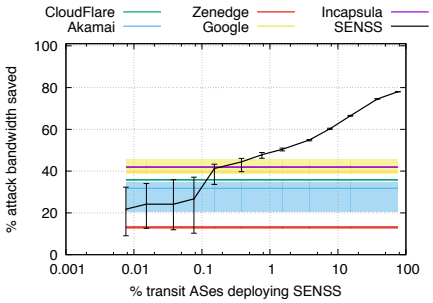
Realistic traffic: We randomly select a victim but distribute attackers at Mirai bot locations, and legitimate clients at large US residential ISPs, as we have done in Section 4.2. We show results only for the top deployment, because the random deployment results do not change with traffic distribution.

We show the median percentage of attack traffic filtered, for flood w sig and reflector attacks (Fig 5(a) and 5(d)), flood w/o sig (Fig 5(b) and 5(e)), and for cross-fire (Fig 5(c) and 5(f)). For flood w/o sig we use $\alpha = 5\%$. Other values of α lower SENSS's effectiveness for small number of deployment points (up to 1,000 ASes) by up to 50%, and we omit them for space reasons. They have no effect for deployments higher than 1,000. For flood w and w/o sig and for reflector attacks we show separately the benefits to direct customers of the SENSS-deploying ISP, and to remote victims that may request SENSS services when under attack. Cross-fire creates disturbance far from the attack's victim, and we show benefit to all ASes.

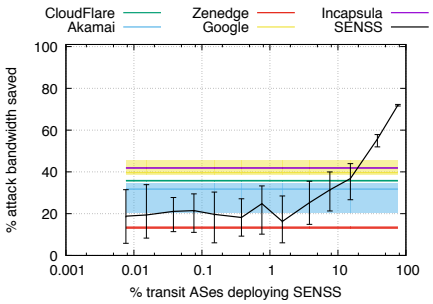
Flood w sig and reflector attacks. In case of these attacks, our results are consistent for both uniform and realistic traffic patterns. Direct, single-homed customers of SENSS ISPs receive almost total protection at all times, both in top and in random deployments, and under both uniform and realistic traffic.

Direct, multi-homed customers receive lower protection than single-homed, because some of their traffic traverses non-SENSS ISPs. During an attack, a multi-homed victim could temporarily become single-homed, to increase its protection. It could do so by selectively announcing its prefixes only to SENSS ISPs.

Remote ASes that request SENSS services (aka "remote customers") receive protection only after a certain number of deployment points is reached. This is where *top* deployment is vastly superior to *random*. Deployment at only 0.7% top ASes achieves 90% protection for remote customers. On the other hand, 64% of randomly chosen ASes must deploy SENSS to achieve 90% protection for remote customers.



(a) Flood w sig – top



(b) Flood w sig – random

Figure 6: Comparison of SENSS versus several cloud-based DDoS defenses, with regard to bandwidth consumed by attack.

Flood w/o signature. In this simulation we assume that SENSS deploys coarse-grained rules, filtering all traffic flowing to the attack's victim at select filter locations. For uniform traffic pattern, SENSS needs large deployment for effective attack mitigation. This is because SENSS must filter close to attack sources to meet the requirement for collateral damage (controlled by $\alpha = 5\%$). At small deployment, SENSS either misses some attack, or its filters would cause too high collateral damage. Deployment at 1.5% top ASes is needed to achieve more than 90% protection for direct, single-homed customers and deployment at 3.8% top ASes is needed for 90% protection for direct, multi-homed and remote customers. Random deployment does much worse against flood w/o sig, where 70% deployment is needed for 90% protection for everyone.

For realistic traffic patterns, SENSS is very effective for direct, single-homed customers, even with 1–10 deployment points (0.01–0.1 on x-axis). This is similar to our result for the 2016 Dyn attack.

Cross-fire attacks. Results are similar for both traffic patterns. *Top* deployment again vastly outperforms *random* deployment. Deployment at the top 1% of ASes can mitigate 90% of cross-fire attacks, while 66% of randomly selected ASes must deploy SENSS to achieve the same effectiveness.

Cloud	ASes	Providers	Peers	Avg. AS-path length
CloudFlare	7	41	185	3.2
Google	20	12	187	2.8
Akamai	38	374	199	3.5
Incapsula	1	69	130	3
Zenedge	1	14	0	4

Table 4: Providers and peers of select clouds, which we use in our evaluation.

4.4 Comparison of SENSS and Cloud Defenses

In this section we compare SENSS's performance to that of clouds, with regard to bandwidth saved during an attack. We assume the same kind of filtering deployed by SENSS and clouds, to isolate the effect of deployment points. This evaluation shows that on-path deployment is superior to the case when traffic is diverted to a cloud.

We first calculate the amount of bandwidth consumed by attack traffic on inter-AS links, as follows. For each of our effectiveness scenarios (Section 4.3) whenever an attack flow crosses an inter-AS link we add its volume to the total consumption. We assume a perfect defense, which drops all attack traffic when it reaches the defense. The difference between bandwidth consumption without and with defense becomes the saved bandwidth. We report it as percentage of the attack bandwidth when there is no defense. An ideal defense would save close to 100% of the bandwidth because it would be deployed close to the sources.

We select five clouds to compare to, which currently offer DDoS defense – CloudFlare, Google, Akamai, Incapsula and Zenedge. While we do not know their peering agreements, we can easily find out their AS ownership from public records. We then obtain peering for those ASes from the CAIDA's AS-level topology. When a cloud owns multiple ASes, we assume that all such ASes deploy the defense. Table 4 shows the number of ASes each cloud owns in our topology, number of providers and peers in our AS topology, and the average AS path length from all other ASes to the cloud.

Figure 6(a) shows the saved bandwidth under top SENSS deployment (median shown with line and 25% and 75% with errorbars), and Figure 6(b) shows it under random SENSS deployment, for flood w sig attack. In both Figures we also show with colored horizontal bars 25% and 75% of bandwidth consumption when the victim is defended by clouds, and the lines show the median of the same measure. Intuitively, bandwidth savings will be largest when attack is filtered close to its sources. There are significant differences among cloud defenses – ranging from 13% bandwidth saved by Zenedge to 38–46% by Google. These differences occur because some clouds have long AS paths (e.g., Zenedge), i.e., they are far from sources, while others (e,g,. Google) have short paths. SENSS outperforms all clouds after 0.4% of top transit ASes (52 ASes, on par with Google's and Akamai's AS count) or after 15% of random transit ASes deploy SENSS. This is because SENSS's on-path defense stops the attack closer to its sources, than when attack traffic must be diverted to clouds.

4.5 Delay, Traffic and Message Cost

All communication between a SENSS client and a SENSS server occurs in one session, over SSL. It takes two round-trip times for SSL establishment. After this, the client would send a query and wait for a reply. Finally it may send a control message to the server to mitigate attacks. Each of the four attack types we studied require 1–3 messages per SENSS ISP for mitigation.

The client can strike the balance between achieving a fast response (ask all SENSS servers for help) and saving money (ask servers one by one). The client can first communicate with Tier-1 and Tier-2 ASes simultaneously, and then switch to iterative communications. This yields on the average 10-second delay, and 300–400 messages for full mitigation.

When using SENSS to mitigate floods w/o sig, the victim must periodically issue traffic queries to learn legitimate traffic distribution, and identify links that carry a lot of legitimate traffic. During attacks the SENSS client uses its most recent observation to estimate collateral damage for a given filter deployment. More frequent queries increase message cost but may reduce collateral damage if traffic fluctuates a lot. We investigated the impact of query period on collateral damage, by replacing legitimate traffic in our effectiveness experiments by traffic volumes and destinations obtained from 24 hours of traffic logs from a large US CDN. We calculated traffic volumes per each minute in the logs and used earlier observations to make SENSS decisions about filter locations, then used later observations to estimate collateral damage. We summarize results of this experiment: (1) observation periods of up to 12 h only slightly increase collateral damage over the ideal case, when we observe immediately prior to attack, (2) filtering at large ASes has higher fluctuation of collateral damage, due to higher traffic aggregation, than filtering at smaller ASes.

4.6 Scalability within an ISP

SENSS functionalities in switches are implemented on the fast path, and incur no per-packet overhead. Further, each SENSS request results in one rule per switch. In our emulation experiments it took only 0.15 sec to handle a single traffic_query and 0.26 sec to handle a route demote. This includes the propagation delay between the victim and the SENSS server (0.05 sec), RPKI validation (0.02 sec), and SENSS processing of the query (0.03 sec). When handling 100 concurrent requests, it took 4.32 s to fully serve traffic_queries, and 24.95 s to serve route_queries followed by demote messages. The delay mostly comes from the concurrent telnet requests to switches, and can be further reduced if we parallelize this communication.

In Figure 7 we illustrate one experiment when flood w/o sig attack is handled by SENSS. The SENSS client at the victim uses our client program from Section 3.4 for handling floods w/o sig. This includes two types of SENSS messages: a traffic_query and a filter. The attack is fully handled within 7 seconds.

5 CONCLUSION

Volumetric DDoS attacks cannot be handled by the victim, because they usually create congestion upstream from the victim. We have proposed SENSS – a framework for collaborative diagnosis and mitigation of volumetric attacks. SENSS's simple but powerful interfaces enable victim-customized solutions to several DDoS variants. SENSS mitigates attacks instead of withstanding them. SENSS is implementable in today's ISPs with SDN, it is very effective in sparse deployment, much faster than manual inter-AS collaboration,

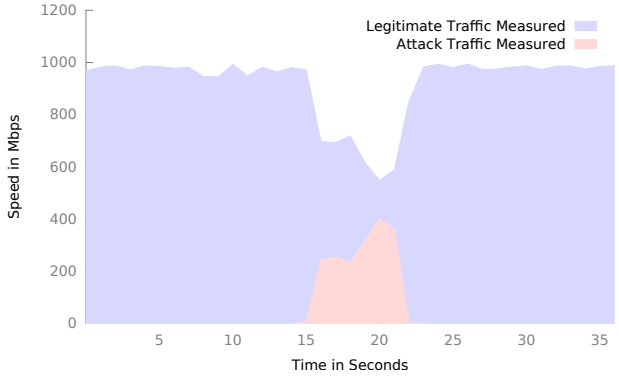


Figure 7: SENSS client using our client program to mitigate floods w/o signature attack.

and has small message overhead. We hope that these good features will encourage its wide adoption.

6 ACKNOWLEDGEMENT

This project is the result of funding provided by the Science and Technology Directorate of the United States Department of Homeland Security under contract number D15PC00184. The views and conclusions contained herein are those of the authors, and should not be interpreted necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security or the US Government. Authors are grateful to anonymous reviewers for their helpful comments.

REFERENCES

[1] Katerina Argyraki and David R. Cheriton. 2005. Active Internet Traffic Filtering: Real-time Response to Denial-of-service Attacks. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC '05)*. USENIX Association, Berkeley, CA, USA, 10–10. http://dl.acm.org/citation.cfm?id=1247360.1247370

[2] R. Bajcsy, T. Benzel, M. Bishop, B. Braden, C. Brodley, S. Fahmy, S. Floyd, W. Hardaker, A. Joseph, G. Kesidis, K. Levitt, B. Lindell, P. Liu, D. Miller, R. Mundy, C. Neuman, R. Ostrenga, V. Paxson, P. Porras, C. Rosenberg, J. D. Tygar, S. Sastry, D. Sterne, and S. F. Wu. 2004. Cyber Defense Technology Networking and Evaluation. *Commun. ACM* 47, 3 (March 2004), 58–61. http://doi.acm.org/10.1145/971617.971646

[3] Sean Bakley. 2017. From Comcast to Hawaiian Telcom: Tracking the top 16 residential broadband service providers in Q3 2017. FierceTelecom, https://goo.gl/otRTw2.

[4] Cristina Basescu, Raphael M. Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. 2015. SIBRA: Scalable Internet Bandwidth Reservation Architecture. *CoRR* abs/1510.02696 (2015).

[5] CAIDA. 2017. The CAIDA AS Relationships Dataset, May 01, 2017. http://www.caida.org/data/as-relationships/.

[6] CloudFlare. 2018. CloudFlare Web page. https://www.cloudflare.com/.

[7] Tim Dierks and Eric Rescorla. 2008. Rfc 5246: The transport layer security (tls) protocol. *The Internet Engineering Task Force* 3 (2008).

[8] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. 2015. Bohatei: Flexible and Elastic DDoS Defense. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 817–832.

[9] Lixin Gao. 2001. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking* 9, 6, 733–745.

[10] Michael T. Goodrich. 2008. Probabilistic Packet Marking for Large-scale IP Traceback. *IEEE/ACM Transaction on Networking* 16, 1 (February 2008), 15–24. https://doi.org/10.1109/TNET.2007.910594

[11] MAWI group. 2017. MAWI Working Group Traffic Archive. http://mawi.wide.ad.jp/mawi/.

[12] John Ioannidis and Steven M. Bellovin. 2002. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2002, San Diego, California, USA*.

[13] Michael G. Kallitsis, Stilian Stoev, Shrijita Bhattacharya, and George Michailidis. 2015. AMON: An Open Source Architecture for Online Monitoring, Statistical Analysis and Forensics of Multi-gigabit Streams. *CoRR* abs/1509.00268 (2015).

[14] Min Suk Kang, Virgil D. Gligor, and Vyas Sekar. 2016. Defending Against Evolving DDoS Attacks: A Case Study Using Link Flooding Incidents. In *Security Protocols Workshop (Lecture Notes in Computer Science)*, Vol. 10368. Springer, 47–57.

[15] Min Suk Kang, Virgil D. Gligor, and Vyas Sekar. 2016. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*.

[16] M. S. Kang, S. B. Lee, and V. D. Gligor. 2013. The Crossfire Attack. In *2013 IEEE Symposium on Security and Privacy*. 127–141.

[17] Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah, and H. J. Chao. 2006. PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks. *IEEE Transactions on Dependable and Secure Computing* 3, 2 (April 2006), 141–155.

[18] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. 2011. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (October 2011), 1765–1775.

[19] Soo Bum Lee, Min Suk Kang, and Virgil D. Gligor. 2013. CoDef: Collaborative Defense Against Large-scale Link-flooding Attacks. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '13)*. ACM, New York, NY, USA, 417–428. http://doi.acm.org/10.1145/2535372.2535398

[20] Xin Liu, Xiaowei Yang, and Yanbin Lu. 2008. To Filter or to Authorize: Network-layer DoS Defense Against Multimillion-node Botnets. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM '08)*. ACM, New York, NY, USA, 195–206. http://doi.acm.org/10.1145/1402958.1402981

[21] MANRS. 2018. MANRS for Network Operators. https://www.manrs.org/manrs/.

[22] P Marques, N Sheth, R Raszuk, B Greene, J Mauch, and D McPherson. 2009. Dissemination of Flow Specification Rules. RFC 5575.

[23] Andrew Mortensen, Flemming Andreasen, Tirumaleswar Reddy, Christopher Gray, Rich Compton, and Nik Teague. 2018. *Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture*. Internet-Draft draft-ietf-dots-architecture-07. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-dots-architecture-07 Work in Progress.

[24] 360.com NetLab. 2017. A quick stats on the 608,083 Mirai IPs that hit our honeypots in the past 2.5 months. https://goo.gl/NYWMLq.

[25] Arbor Networks. 2018. DDoS Protection by Arbor Networks APS. https://www.arbornetworks.com/ddos-protection-products/arbor-aps.

[26] George Oikonomou, Jelena Mirkovic, Peter Reiher, and Max Robinson. 2006. A Framework for a Collaborative DDoS Defense. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference*. IEEE Computer Society, 33–42.

[27] Vern Paxson. 1999. Bro: A System for Detecting Network Intruders in Real-time. *Comput. Netw.* 31, 23-24 (December 1999), 2435–2463. http://dx.doi.org/10.1016/S1389-1286(99)00112-7

[28] Adrian Perrig, Pawel Szalachowski, Raphael M. Reischuk, and Laurent Chuat. 2017. *The SCION Architecture*. Springer International Publishing, Cham. 17–42 pages. https://doi.org/10.1007/978-3-319-67080-5_2

[29] Steve Ranger. 2018. GitHub hit with the largest DDoS attack ever seen. ZDNet,https://goo.gl/BmqekG.

[30] Matthew Roughan. 2005. Simplifying the Synthesis of Internet Traffic Matrices. *SIGCOMM Comput. Commun. Rev.* 35, 5 (October 2005), 93–96. https://doi.org/10.1145/1096536.1096551

[31] The New York Times. 2013. How the Cyberattack on Spamhaus Unfolded. http://www.nytimes.com/interactive/2013/03/30/technology/how-the-cyberattack-on-spamhaus-unfolded.html.

[32] D. Turk. 2004. *Configuring BGP to Block Denial-of-Service Attacks*. RFC 3882. RFC Editor.

[33] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. 2010. DDoS Defense by Offense. *ACM Trans. Comput. Syst.* 28, 1, Article 3, 54 pages. http://doi.acm.org/10.1145/1731060.1731063

[34] X. Yang, D. Wetherall, and T. Anderson. 2008. TVA: A DoS-Limiting Network Architecture. *IEEE/ACM Transactions on Networking* 16, 6 (Dec 2008), 1267–1280.

[35] Kyle York. 2016. Dyn Statement on 10/21/2016 DDoS Attack. https://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/.

[36] Zenedge. 2018. Zenedge Web page. https://www.zenedge.com/.