Computer Communications 150 (2020) 438–454

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Net-Police: A network patrolling service for effective mitigation of volumetric DDoS attacks

Sareena Karapoola *, Prasanna Karthik Vairam, Shankar Raman, V. Kamakoti

Department of Computer Science & Engineering, Indian Institute of Technology Madras, Chennai, India

ARTICLE INFO

Keywords:
DDoS defense
Patrol algorithm
Packet tagging
IP traceback

ABSTRACT

Volumetric Distributed Denial of Service (DDoS) attacks are a significant concern for information technology-based organizations. These attacks result in significant revenue losses in terms of wastage of resources and unavailability of services at the victim (e.g., business websites, DNS servers, etc.) as well as the Internet Service Providers (ISPs) along the path of the attack. The state-of-the-art DDoS mitigation mechanisms attempt to alleviate the losses at either the victim or the ISPs, but not both. In this paper, we present Net-Police, which is a traffic patrolling system for DDoS mitigation. Net-Police identifies the sources of attack so that filters can be employed at these sources in order to quickly mitigate the attack. Such a solution effectively prevents the flow of malicious traffic across the ISP networks, thereby benefiting the ISPs also. Net-Police patrols the network by designating a small number of routers as dynamic packet taggers, to prune benign regions in the network, and localize the search to the Autonomous Systems (AS) from which the attack originates. We evaluate the proposed solution on 257 real-world topologies from the Internet Topology Zoo library and the Internet AS level topology. The paper also presents details of our hardware test-bed platform consisting of 30 routers on which network services such as Net-Police can be implemented and studied for on-field feasibility. Our experiments reveal that Net-Police performs better than the state-of-the-art cloud-based and traceback-based solutions in terms of ISP bandwidth savings and availability of the victim to legitimate clients.

1. Introduction

With the growing intervention of Information Technology based-services into every area of business, ensuring the uninterrupted availability of these services is of prime importance. Volumetric Distributed Denial of Service (DDoS) attacks pose a big challenge in this context. Examples of such volumetric DDoS attacks include the Mirai Botnet attack [1] on Dyn DNS service and the attack on BBC [2]. Typically, DDoS attacks employ a large number of bots and/or amplifiers, thereby resulting in a flood of untraceable attack traffic towards the victim [1,3,4]. These attacks not only result in reputation loss caused by non-availability of service but also significant revenue loss [5].

A victim of the DDoS attack is affected by the loss of revenue, reputation due to its downtime [6], and also due to the significant cost involved in mitigating the DDoS attacks [5]. At the same time, DDoS attacks are also detrimental to the Internet Service Providers (ISPs) along the path of the attack traffic. The huge volume of attack traffic forwarded by various ISPs along the path result in a significant amount of wasteful resources (computation and memory) being used at their routers. This in-turn results in: **(1)** financial loss from the wastage of bandwidth due to attack traffic; and, **(2)** reputation loss due to both transgression of peering agreements [7] and degradation in Quality of Service (QoS) offered to legitimate users [8]. Also, the revenue loss due to DDoS can potentially eclipse the revenue gain from long-term business strategies of the ISPs such as the flat-rate pricing plan.[1] Existing DDoS mitigation mechanisms attempt to alleviate the issues mentioned above experienced either by the victim or the ISPs, but not both.

Most DDoS mitigation schemes used in practice attempt to filter the DDoS traffic at either the first-hop ISP [7] (e.g., Blackholing) or a cloud-based traffic scrubbing [9,10] center. These schemes are widely used due to the ease with which the victim can avail these services. However, in the former, a significant number of legitimate clients may not be able to reach the victim even after mitigation, while in the latter, the entire traffic is susceptible to eavesdropping by the third-party scrubbing service center [11]. Furthermore, in both the cases, the attack traffic still flows from the attack source to the point of filtering (i.e., the first-hop ISP or the scrubbing center), thereby wasting bandwidth at the corresponding ISP links. Although practical, the aforementioned

* Corresponding author.
E-mail addresses: sareena@cse.iitm.ac.in (S. Karapoola), pkarthik@cse.iitm.ac.in (P.K. Vairam), mjsraman@cse.iitm.ac.in (S. Raman), kama@cse.iitm.ac.in (V. Kamakoti).

[1] Flat-rate pricing refers to a fixed fee charged for network access, irrespective of the usage volume.

https://doi.org/10.1016/j.comcom.2019.11.034
Received 31 May 2019; Received in revised form 21 September 2019; Accepted 23 November 2019
Available online 4 December 2019

defenses are not designed to benefit the ISPs and may not even benefit the victim completely.

An alternative defense strategy is the one based on traceback, wherein the source of the attack is identified (by the victim, ISP, or both) in order to deploy traffic filters at the ISPs. Traceback-based schemes benefit the ISPs — by ensuring that attack traffic does not transit their links, and they benefit the victim — by ensuring that the victim is reachable to the legitimate clients. However, all the traceback-based solutions [12–18] have one or more of the following drawbacks: **(1)** they lack insights on which routers can be leveraged for traceback [12–16]; **(2)** the computation power required for accurate DDoS detection as well as mitigation is high [16,17]; **(3)** light-weight schemes are inaccurate and end up misleading traceback [12,14]; **(4)** malicious adversaries may interfere and mislead the traceback [12–16]; **(5)** schemes based only on the traffic-volume are light-weight but could be easily misled by the DDoS bots [18]; and, **(6)** they are not compatible with existing protocols[2] [12,13,15–17]. Therefore, there is a need for a traceback-based scheme that is light-weight at both the victim as well as the ISP, in addition to being resilient to adversaries.

In this paper, we propose *Net-Police*, which is a traceback-based mitigation scheme for volumetric DDoS attacks that aims to benefit both the victim as well as the ISPs. Net-Police is a light-weight traffic patrolling system that reliably identifies the sources of attack packets within a short period of time. Net-Police is provided as a *service* by a Central Authority (CA) to an ISP and in-turn to the businesses hosted within the ISP. This service can be triggered either by the ISP itself or the affected business organization when an attack is detected. For each service request, Net-Police *patrols*[3] the network to prune out benign regions and localize the search to the autonomous systems (ASes) from which the attack originates. For this, Net-Police takes the help of a few strategic routers. These routers, which are designated as *packet taggers* (PT), tag packets destined to the victim with a label for a small duration of time. The labels, collected from the attack packets at the victim, assist the CA to prune off benign regions, and identify the sources of the attack. Note that these strategic routers can be selected based on metrics such as centrality, traffic flow rate, or subtree size. For compatibility, Net-Police uses the well-known and well-supported Generic Routing Encapsulation (GRE) tunnels to carry the labels to the victim. After finding the attack sources, the CA can curb the DDoS traffic as per the mitigation policies and the extent of its control over the routers. For instance, the CA can instruct the routers to blackhole [7] or rate limit [19] the attack traffic.

Net-Police is resilient to adversarial attempts by a PT or any intermediate router between the PT and the victim. Such malicious PTs or routers may sniff, spoof, or tamper with the labels in order to misguide and delay the mitigation service. To prevent such attacks, Net-Police employs security mechanisms based on *hopping labels* and encryption. These mechanisms result in overheads at the router. However, these overheads are only incurred at a few routers that participate in packet tagging and that too only for a brief period of time (< 350 ms), unlike [16,17].

The following are the main contributions of this paper.

1. Ours is one of the first work to address the need for a DDoS defense technique that is beneficial to both the ISPs and the victim for it to be commercially viable. As compared to the prior state-of-the-art DDoS mitigation mechanism [18], Net-Police **(i)** ensures availability of services to 92% of the legitimate clients as compared to SENSS [18] which provides 5% availability when the attackers' position is not restricted; and, **(ii)** saves bandwidth at the ISP links by more than 99% as compared to SENSS [18] that can save up to 42% (Section 6.1.2).
2. We provide a real-world implementation of a packet tagger device using off-the-shelf hardware. We also demonstrate the practical feasibility of Net-Police using a physical test-bed created using 30 off-the-shelf devices.
3. We evaluate Net-Police in 257 real-world topologies [20,21], and against real-world attack scenarios such as the Mirai botnet [1,22]. We also determine the most suitable network metric that can facilitate quicker traceback in any network topology.
4. Our results indicate that Net-Police can identify all attack-sources[4] even in a large scale Autonomous Systems (AS) level topology of 63463 nodes in ≈ 3.6 s.

The rest of the paper is organized as follows. Section 2 discusses the literature related to our work. Section 3 presents the proposed solution. Section 4 details the strategic placement of packet taggers. Section 5 presents the adversary model specific to Net-Police and discusses the secure packet tagging procedure. Section 6 presents the evaluation of our solution. We further present a discussion on practical issues related to Net-Police in Section 7. Finally, Section 8 concludes the paper.

2. Related work

In this section, we compare our solution against prior works that have goals similar to ours: mitigation of volumetric DDoS attacks. These works work by filtering the attack packets in the network at the ISP [7], traffic scrubbing centers [9,10,23,24], or much closer to the sources of attack [12–18]. A deeper understanding of these techniques led us to analyze who exactly these schemes benefit.

The benefits of a DDoS mitigation scheme and the cost incurred differ for each stakeholder, namely the victim and the various ISPs. The victim expects that the mitigation cuts off the attack traffic, while its services are available to all its legitimate clients. Further, it expects that the scheme to be light-weight and not compromise the privacy of its business data. On the other hand, the ISPs expect that the mitigation scheme minimize the wasted-bandwidth on its network links due to transit. At the same time, for economic viability, the scheme should be light-weight, compatible with existing protocols, and monetizeable. We observe that almost all existing techniques are beneficial to either the ISP or the victim but do not aim to benefit both.

Table 1 provides a comparison of the various DDoS mitigation schemes in terms of the properties as mentioned above. A scheme is victim-centric or ISP-centric depending on if it benefits the victim or the ISP. Note that an ideal scheme is beneficial to both the victim and all the ISPs on the path from the attack source to the victim [25]. We now review the three classes of works closest to ours along the aforementioned verticals.

Blackholing [7]. In this technique, the first-hop ISP of the victim employs BGP blackholing to ensure that the victim is unreachable from the rest of the Internet, based on the victim's request. It does so by announcing a BGP blackhole route corresponding to the victim, to all its peers and providers. While the victim is relieved of the attack, this scheme has the side-effect of making the victim unreachable to even its legitimate users. In effect, the scheme saves bandwidth only at the first-hop ISP of the victim and the victim itself, but not any other on-path ISP since they still have to carry the attack traffic.

Cloud-based Schemes [9,10,23,24]. In this technique, all the victim-bound traffic is redirected to third-party traffic scrubbing infrastructures that filter/absorb the attack traffic, before tunneling the clean traffic back to the victim. These highly provisioned defenses employ deep-packet inspection, and are scalable to handle volumetric attacks, thereby benefiting the victim. However, the victim still faces privacy issues since their traffic has to transit a third-party infrastructure [11]. However, these techniques do not aim to benefit the ISPs much since

[2] The schemes use well-known fields of the IP headers that can affect compatibility.

[3] Unlike a real-world patrol, Net-Police performs a reactive (not proactive) patrol.

[4] The AS (or its egress router) that hosts the attacker-controlled compromised machines.

Table 1
Comparison of DDoS mitigation schemes.

	Stake-holder	Desired properties	Ideal	Cloud based [10,23,24]	Blackhol-ing [7]	IP-Traceback based			SENSS [18]	Net-Police
						PPM [12, 14,15]	DPM [13, 16]	MOD [17]		
Benefits	Victim	Availability	✓	✓	✕	✓	✓	$	$	✓
		Saves BW	✓	✓	✓	✓	✓	✓	✓	✓
	ISP	Saves BW	✓	✕	✕	✓	✓	*	*	✓
Cost of mitigation	Victim	Privacy compromise of business data	✕	✓	✕	✕	✕	✕	✕	✕
		Overhead	Low	NIL	NIL	High	Low	Low	Low	Low
	ISP	Overhead	Low	High	NIL	Low	High	High	High	Low
		Compatibility	✓	✓	✓	✕	✕	✕	✓	✓
		Scalable	✓	✓	✓	✓	✕	✓	✕	✓
Scheme is centric to			On-path ISPs and victim	Victim	First ISP	Victim	Victim	Victim	Victim	On-path ISPs and victim

✓ Yes.
✕ No.
* ISP Bandwidth is partially saved.
$ Victim is available to a fraction of its clients.

they still have to carry the high-volume attack traffic to the mitigation centers.

IP Traceback-based schemes [12–17]. This technique works by identifying the on-path routers preferably closest to the attack sources, unlike blackholing or cloud-based schemes, for filtering attack packets. This strategy benefits both the victim and the ISPs when implemented correctly. While many approaches to IP traceback have been proposed, the notable ones are the solutions based on packet marking.

In packet marking schemes, the ISP routers mark their unique identities, called a *label*, in the IP headers of the packets passing through them, while the recipient analyzes these labels in the packets to retrace the path to the attacker *hop-by-hop* until the attacker is found [13]. Subsequently, this knowledge of the attacker's location is used for installing filters at a router that is closest to the attack source. Such a scheme effectively relieves the victim and all the ISPs on the path from having to deal with attack traffic. Note that the labels can be inserted either probabilistically (PPM) [12,14,15] or deterministically (DPM) [13,16,17] by the ISP routers. While the former falls short in benefitting both ISP and victim, due to the inaccurate tracebacks resulting from its probabilistic nature, the latter poses significant challenges for deployment at the ISPs. Specifically, some of the DPM schemes [16] incur a considerable loss of network throughput at the routers due to their always-on marking. On the other hand, a recent DPM scheme, namely MOD [17], attempts to overcome the drawbacks of always-on marking but requires DDoS detection modules at the ISP routers, thereby adding to the cost. Another issue with existing packet marking schemes in general is the compatibility issues with the existing protocols on the Internet.

SENSS [18], the work that is closest to ours, also aims to benefit both the victim and the ISPs. To make the traceback light-weight and compatible for the ISPs, unlike packet marking, SENSS relies on traffic statistics to ascertain the source of the attack. SENSS stops the attacks by choosing to install a filter at a router that forwards the maximum volume of traffic towards it. Note that the victim AS queries each router in the internet periodically, hop-by-hop, to get the exact volume of traffic that the router forwards towards it. However, this scheme has the following major drawbacks: **(1)** querying each router on the Internet has non-trivial network overhead; **(2)** maintaining a traffic summary for each victim AS on the Internet incurs significant processing and storage at a router; **(3)** the volume of traffic forwarded by a router may not be indicative of the presence or absence of an attacker behind the router, especially in case of a DDoS. In order to arrive at a practicable solution, SENSS restricts its queries/stats counter deployments to only the Tier-1 and Tier-2 ISPs. Consequently, this may result in inaccurate identification of the source of the attack. Specifically, the attack traffic flowing from the attack source to the Tier-1/2 filtering router cannot be blocked by SENSS in its current form.

In this work, we propose a traceback-based approach that is beneficial to both the victim and the ISPs. Unlike SENSS and the prior traceback-based approaches, our solution: **(1)** uses only a *few* select routers in the network *on-demand* for reducing the overheads of traceback; **(2)** does not require any state/summary maintained at the router; **(3)** can achieve quick traceback unlike the hop-by-hop traceback proposed in earlier works; and, **(4)** exploits well-supported functionalities in the ISP routers to ensure compatibility with existing protocols.

Comparison with SDN-based schemes. Although Net-Police is primarily designed for Internet-scale topologies, it has a few similarities to the traceback mechanisms employed in the domain of Software Defined Networking (SDN) [26,27]. For instance, Net-Police requires a centralized controller (i.e., the CA) as well as uses a protocol similar to OpenFlow for communicating control information to the routers. However, there are significant differences between the two. In the SDN literature [26,27], the controller utilizes its knowledge of the OpenFlow rules at the routers to eventually identify the source. Extending such solutions to the Internet is close to impossible since the Internet routers are spread across multiple administrative and geographical domains, and the ISPs may not willingly share privileged information such as router-level paths [28]. In contrast, Net-Police does not require such privileged information for its functioning but instead requires only the BGP-level topology [29,30].

3. The proposed solution: Net-police

In this section, we present a high-level overview of Net-Police. Let $G = (V, E)$ represent the topology of the Internet with $|V|$ nodes and $|E|$ links as shown in Fig. 1a. Each node denotes an Autonomous System (AS) that is owned by an ISP. Note that the AS can and most likely will contain multiple physical routers that are not shown in the figure. Each edge denotes a network link between the egress router of an AS to the ingress router of another AS. Attackers (bots) behind an unknown and possibly distributed set of ISP ASes send malicious attack traffic to a victim $v \in G$. In reality, while the bots can be millions in number, the number of ISP routers behind which these bots are distributed are lesser and are confined to a few organizations belonging to a few countries [31]. Let A be the set of such ASes. We call A as *attack-source* ASes or routers as they knowingly or unknowingly host these bots. Note that DDoS attacks are of two types: **(1)** direct DDoS, wherein the bots send a large volume of traffic directly to the victim [1,32]; and (2) indirect/amplification DDoS, wherein the bots impersonate victims and send requests to open services (e.g. DNS resolvers) such that the

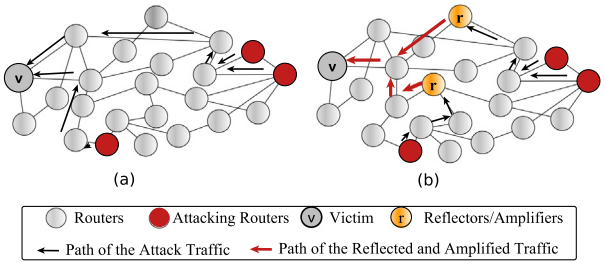

Fig. 1. The AboveNet topology [21]. (a) Direct DDoS attack; and (b) Indirect DDoS attack, on v.

responses cause a DDoS at the victim [3,4]. As shown in Fig. 1a, in the case of direct attacks, v receives traffic directly from A, whereas in the case of amplification attacks (Fig. 1b), v receives the response traffic from the open servers also called as reflectors r (for the requests from A). The objective of this work is to determine the set A efficiently and stop the flow of malicious packets. Efficiency is characterized by how quickly the objective is met with the least consumption of **(i)** bandwidth on the links; and, **(ii)** compute resources on the routers.

The proposed technique assumes a Central Authority (CA), which serves as a trusted entity that is responsible for patrolling the network. The CA runs the Net-Police solution to identify the attack sources and initiate curbing of malicious traffic. The CA requires the co-operation of a large number of ISPs, realizing which may not be easy. Inspired by solutions in other domains [33–35], we promote Net-Police as a consortium of ISPs rather than a separate commercial entity [36]. Consequently, the CA has knowledge of the AS-level topology [29,37] and can initiate and stop packet tagging at the egress router of the participating ASes. It can also issue commands to such routers through a dedicated communication channel that is minimalistic and secure. Such dedicated channels between the CA and the participating routers can be established using any secure communication protocol such as those reported in literature [38]. *As the channel is dedicated, the CA becomes accessible to all the participating ASes even during fatal volumetric DDoS attacks.* The bandwidth requirement of this dedicated channel in the context of Net-Police is less as estimated in Section 5.3.1 of this paper.

We describe the working of Net-Police in Algorithm 1, using the example of a *direct attack* on v as shown in Fig. 2. We also show that the same steps are easily adaptable to amplification attacks next.

In the case of an amplification DDoS attack, Net-Police will traceback to the reflector and not the actual source of the attack. This is because the source IP addresses in the malicious packets at v are of the reflectors. However, Net-Police can traceback to the actual source with minimal effort from the reflectors as we show next. To achieve this, the CA must first know that the attacker is using a reflector — either from the reflector itself (using its own DDoS detection mechanisms) or the victim (after a traceback). Once the attack is reported, Net-Police will consider the reflector as the new victim and start a new patrol. To participate in Net-Police operations, the reflector needs to support a few basic protocols that are both light-weight and readily available in almost all off-the-shelf routers (refer Section 7). Note that most reflectors will be willing to participate in Net-Police especially since they are maintained by the ISPs themselves [1] and it does not require much effort. Nevertheless, there can be some reflectors that may not cooperate. In such a case, the best that Net-Police can offer is to localize the attack source to a router close to the reflector. Tracing back to the actual attack sources of such reflection attacks require further effort, which we plan to address in a future work.

We now discuss some aspects of implementing Net-Police in practise. It is straightforward to note that packet tagging is enabled on any router at most once. This ensures that Net-Police will terminate in a

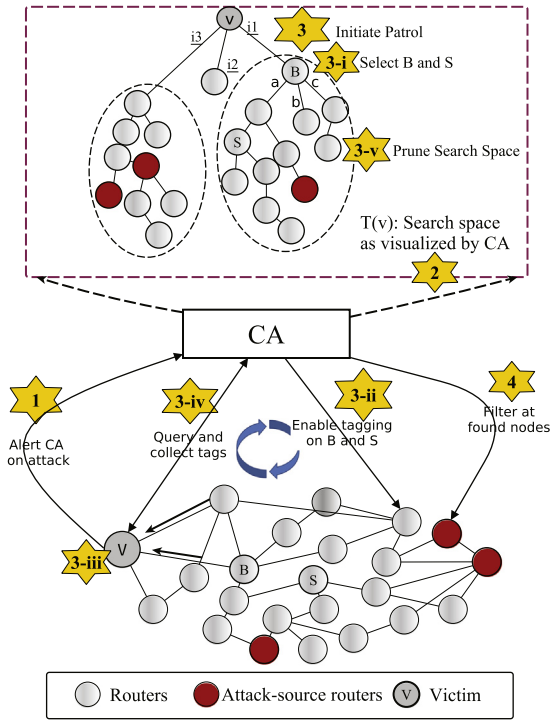

Fig. 2. Illustration of the working of Net-Police..

bounded time. The efficiency of the scheme is measured by how *quickly* and *accurately*, the step 3 explores (prunes) the tree $T(v)$ to converge on the attack-source routers. In this context, the two main challenges faced by Net-Police are — **(1)** optimal *trade-off* between quickness and overhead; and, **(2)** ensuring the *correctness* of the traceback.

The convergence time (i.e. the number of cycles required to explore the tree $T(v)$) is determined by the *number of PTs* that can be *simultaneously* enabled in a subtree-of-interest. If only one router is enabled as a PT, then step 3 would proceed *hop-by-hop* to prune $T(v)$, thereby increasing the convergence time. On the other hand, if all routers are enabled as packet taggers, Net-Police can easily converge on the attack-source routers in just one cycle. However, when all routers start tagging packets, in addition to the efforts incurred at the routers, the packets also increase in size. This is because, every PT *appends*[5] its label to the header of the packet flowing through it. The resulting bandwidth wastage on the links carrying the labeled traffic, coupled with the resource utilization at these routers, causes a trade-off between convergence time and the overheads incurred in packet tagging. We discuss this trade-off in Section 4, specifically highlighting how an optimal choice of a small number of PTs can prune more nodes per cycle for quick convergence.

The *correctness* of the proposed solution crucially depends on the requirement that the labels cannot be tampered with. Since the labels are the main *diagnostic tool* of Net-Police, a tampered/spoofed label can potentially *mislead* the CA. Further, we assume a strong adversary model, wherein any router in the network can be compromised, and the adversary may also be aware of the working of Net-Police. Given this adversary model, there is a need to have a secure tagging mechanism, that maintains the integrity of the labels while the packet flows from the PT to the victim. Section 5 presents details of the secure tagging mechanism adopted in Net-Police.

[5] When the packets flow through PTs $1..s$, they carry the labels of the corresponding PTs concatenated in their headers. i.e $M_1 \parallel M_2 \parallel \ldots \parallel M_s$.

Algorithm 1: NET-POLICE

1. The action starts when victim v detects a DDoS attack on it and identifies the malicious packets. There is a wide range of schemes that v can employ for identification of malicious packets, starting from simple techniques like threshold violation by traffic flows to using sophisticated schemes that identify the attack packets using signatures [13,15].
2. The victim v alerts the CA about the attack by providing the signature of the attack and the ingress interfaces on which it received the packets. As seen in Fig. 2, the CA *visualizes* the search space as a tree $T(v)$ rooted at the victim v, that shows all active AS-level paths from the Internet ASes to the victim. Note that these routes do not frequently change over time and can be obtained very easily in our consortium model [15,29,37]. As the CA is aware of the interfaces on v that is attacked, it identifies the subtrees-of-interest. As seen in Fig. 2, the subtrees of $T(v)$ attached to v through interfaces i_1 and i_3 are of interest.
3. In parallel, for each subtree-of-interest (attached to v through the respective interfaces), the CA performs the following *patrol* steps. Fig. 2 illustrates the same on the subtree attached on interface i_1 of v.
   - i The CA *selects* a set of ASes as *checkpoints* that shall assist in identifying benign and malicious subtrees in $T(v)$. The selection of the checkpoints is based on the metrics defined in Section 4.2. In this example, the ASes B and S are selected as checkpoints. These checkpoints, also called as *packet taggers* (PTs), ensure that any victim-bound packet crossing it is tagged with a label. Such a checkpoint AS may internally choose any of its appropriate egress routers to ensure the same. Since a PT is deployed only at the egress router of the AS, henceforth in the paper, we use AS and router interchangeably to denote a node in $T(v)$.
   - ii The CA enables packet *tagging* at these selected routers for a *pre-determined time period* t. Accordingly, these PTs shall *tag* packets with a *label* that is not spoofable and further unique to each PT. The label, which is denoted by M, is composed by concatenating a unique router identifier RID and the ingress interface identifier (I) through which the packet arrived. The time period t is pre-determined based on the geographical/logical distances between the routers, to ensure that at least one tagged packet from the PT reaches the victim. Since most destinations across the world have a round trip time < 350 ms [39], t is estimated to be ≤ 350 ms.
   - iii The victim v receives labeled packets through interface i_1 including the malicious ones by the time period t as defined in step 3[ii].
   - iv The malicious packets in this example will have *labels* (of the form M_{B_a}) tagged by B, with the interface a of B through which the packets arrived. Note that these packets will not have the label of S, as they do not pass through S. This information is collected by CA by querying v.
   - v Based on the same, the entire subtree rooted at S, represented by $T(S)$, in Fig. 2 is inferred as *benign*, and is *pruned* (certified to not contain the malicious router and hence not explored further) from $T(v)$. Further, the interface identifier a in the received label (M_{B_a}) assist CA to infer that the subtree attached on interface a at B is malicious. Accordingly, the other interfaces at B (b and c) are inferred as benign, and the subtrees attached on these interfaces are pruned as well. Additionally, if the interface identifier in the label is an internal interface of a PT, Net-Police identifies the PT as the *source* of the attack packet.
   - vi At the end of the time-period t, the CA disables packet tagging at the PTs B and S.

   The steps 3[i] to 3[vi] are referred to as a *patrol cycle*. The malicious subtrees not pruned in $T(v)$ form the subtrees-of-interest for the next cycle. The cycles are continued on the rest of the search space, either until the CA finds all the attack-source routers (i.e. explores the entire tree $T(v)$) or the victim requests to stop.
4. Once the CA identifies the attack sources, it initiates filtering at the nearest router which it can control.

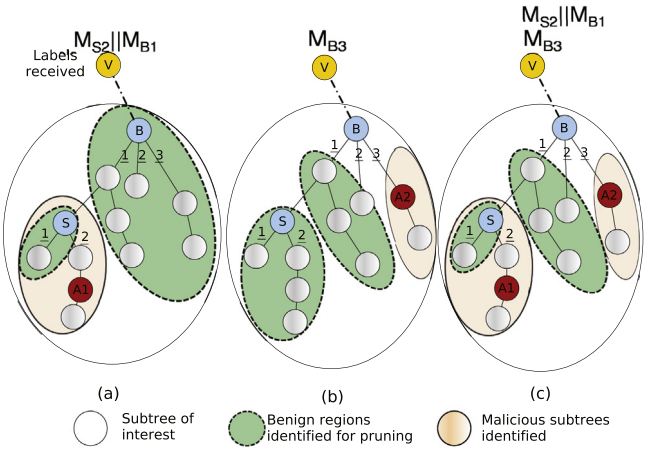

Fig. 3. Illustration of pruning in Net-Police.

4. Pruning using packet taggers

As mentioned earlier, the challenge is to select a minimum number of Packet Taggers (PTs) that could prune the maximum number of nodes thus enabling quick convergence. For a better understanding of this selection process, we first discuss how these PTs assist the Central Authority (CA) to localize the search by *pruning* benign regions. Subsequently, we propose a set of *selection metrics*, which can be used to identify such strategic routers in each patrol cycle of Net-Police.

4.1. Pruning

Initially, the first patrol cycle of Net-Police starts with subtrees-of-interest directly attached to the victim v in $T(v)$. As the patrolling proceeds, each cycle prunes benign regions from these subtrees and generates multiple subtrees-of-interest for subsequent cycles. For each subtree-of-interest, in parallel, the proposed solution uses two PTs B and S. While the root of the subtree-of-interest is chosen as B, a node inside this subtree is chosen as S. Empirically, it is shown that two simultaneous PTs *per subtree-of-interest* is sufficient for quick convergence (Refer Section 6) while limiting the overhead incurred within acceptable limits (Refer Section 5.3.3).

Fig. 3 illustrates different scenarios that arise when the CA selects two nodes B and S as PTs in a subtree-of-interest. The larger white circle is the initial subtree-of-interest. In Fig. 3(a) attackers are in the subtree attached to the interface 2 of S. So the malicious packets received by v will have the label $M_{S_{I_2}} \parallel M_{B_{I_1}}$ tagged by both S and B. The CA uses these labels to prune the subtrees-of-interest for subsequent patrol cycles. In this case, the CA prunes all the nodes and links marked red in Fig. 3(a) and uses the remaining part for

Algorithm 2: ANALYZE_LABELS

input : $\mathbb{T}$: Subtree-of-interest $\mathbb{L}$, Set of labels received by victim v in malicious packets originating from $\mathbb{T}$

output: $\mathbb{BR}$: Benign Regions identified; $\mathbb{SI}$: Subtrees-of-Interest for the next cycle; $\mathbb{FL}$: Found list of attack-source routers.

1 /*$B_{I_{i=0..d_B}}$ and $S_{I_{k=0..d_S}}$ denote interfaces of B and S respectively. Let $B_{I_{i=j}}$ represent the interface of B under which S is present. The packets flowing through B and S are tagged with the corresponding labels $M_{B_{I_i}}$ and $M_{S_{I_k}}$. Based on the labels following benign regions are identified:

2 **Region i:** Subtrees rooted at benign interfaces of B

3 **Region ii:** Subtrees rooted at benign interfaces of S

4 **Region iii:** Nodes in between B_{I_j} and S in $T(B_{Ij})$*/

5 **Scenario 1: if** $M_{S_{I_k}} \| M_{B_{I_j}}$ *alone is in* $\mathbb{L}$ **then**

6 /*Attackers are present only in $T(S)$.*/

$\mathbb{SI}$ = $\{T(S_{I_k})\}$

7 $\mathbb{BR}$ = $(T(B) - T(S))$ /* **Regions i and iii***/

$\cup\ (T(S) - T(S_{I_k}))$ /* **Region ii***/

8 **end**

9 **Scenario 2: else if** $M_{B_{I_i}}$ *alone is in* $\mathbb{L}$ **then**

10 /*Attackers are present in $T(B)$-$T(S)$, not in $T(S)$*/

$\mathbb{SI}$ = $\{T(B_{I_i})\}$

11 $\mathbb{BR}$ = $(T(B) - T(S))$ /* **Regions i and iii***/

$\cup\ (T(S) - T(S_{I_k}))$ /* **Region ii***/

12 **end**

13 **Scenario 3: else if** *Both* $M_{S_{I_k}} \| M_{B_{I_j}}$ *and* $M_{B_{I_i}}$ *are in* $\mathbb{L}$ **then**

14 /*Attackers are present in both $T(S)$ and $T(B)$-$T(S)$. */

$\mathbb{SI}$ = $\{T(S_{I_k}), T(B_{I_i})\}$

15 $\mathbb{BR}$ = $(T(B) - T(B_{I_i}))$ /* **Region i***/

$\cup\ (T(S) - T(S_{I_k}))$ /* **Region ii***/

16 **end**

17 /*If the label indicates a local interface at a PT, the PT is inferred as the source of attack.*/

18 **for** l *in* $\mathbb{L}$ **do**

19 $(RID, I) \leftarrow$ Extract router identifier and interface from l

20 **if** I *is an internal interface of* RID **then**

21 $\mathbb{FL} \cup= RID$

22 **end**

23 **end**

exploration in subsequent cycles. In Figs. 3(b) and 3(c), the malicious packets received at v shall have labels $M_{B_{I_3}}$, and a combination of $M_{S_{I_2}} \| M_{B_{I_1}}$ and $M_{B_{I_3}}$, respectively resulting in the corresponding pruning as shown.

For each subtree-of-interest $\mathbb{T}$, Algorithm 2 generalizes the pruning done by the CA by analyzing the labels received in malicious packets at v, from the attackers in $\mathbb{T}$ during each cycle of Net-Police. Such labels will have $RID \in \{B, S\} \subset \mathbb{T}$. The algorithm outputs $\mathbb{BR}$, the set of benign nodes identified for pruning in $\mathbb{T}$; $\mathbb{FL}$, the attacking nodes found in $\mathbb{T}$; and $\mathbb{SI}$, the set of subtrees-of-interest for the next cycle. The completeness of this approach follows from the fact that all the malicious packets will surely be tagged with a label of the form M_{B_I}. Given this, the possible scenarios are whether the packets have labels with **(1)** concatenation of $M_{S_I} \| M_{B_I}$ alone; **(2)** M_{B_I} alone; or, **(3)** both. All of these possible scenarios are addressed in Algorithm 2. Subsequent to pruning, if there exists a label of the form (RID, I) from $RID \in \{B, S\}$, wherein I is an internal interface, then the attack is originating from the RID and would be included in the $\mathbb{FL}$ set.

The selection of S in a subtree-of-interest can impact the number of nodes pruned per cycle and in turn the convergence time of Net-Police. Fig. 4 illustrates two scenarios, wherein the selection of S as in Fig. 4a results in more number of nodes being pruned as compared to that in Fig. 4b. Therefore, the *selection* metric employed by CA to select S is important. We next discuss why the use of two PTs is advantageous for

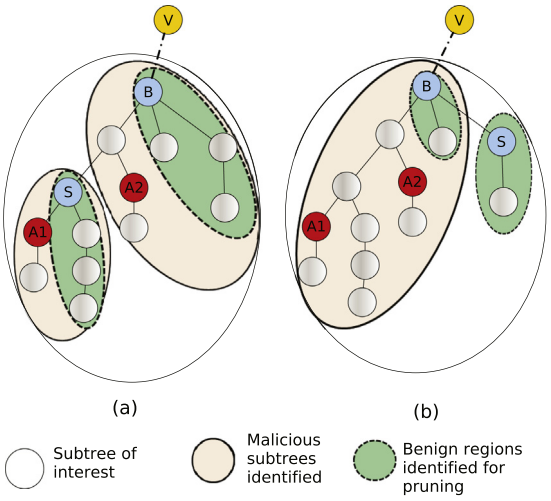

Fig. 4. Optimal selection of S. (a) The placement of S prunes 8 nodes. (b) A sub-optimal placement of S that prunes only 4 nodes.

quick convergence and how the placement of S can impact the number of nodes pruned per cycle.

Using two PTs is Advantageous. Given a subtree-of-interest $\mathbb{T}$, we first analyze the simpler scenario, where only one router is designated as a PT per cycle. The chosen PT B is always the *root* of $\mathbb{T}$. This is done in order to identify the interface(s) from which the attack originates. Let $B_{I_1}, B_{I_2}, \ldots, B_{I_{d_B}}$ be the interfaces of the PT B, where d_B is the degree of B. We denote the tree rooted at B as $T(B)$. The $|T(B)|$ denotes the number of nodes in $T(B)$. Similarly, the subtree attached to an interface B_{I_i} is denoted as $T(B_{I_i})$. If the attack-source routers are behind an interface B_{I_i}, where $1 \le i \le d_B$, then at least one of the attack packets received will have the label $M_{B_{I_i}}$. The function $tag(M_{B_{I_i}})$, computed by Net-Police, returns 1 if the victim receives the label $M_{B_{I_i}}$ in at least one of the attack packets, and 0 otherwise. When $tag(M_{B_{I_i}})$ is 0, the interface B_{I_i} is inferred as benign, and the subtree $T(B_{I_i})$ attached to it can be pruned. Note that B can be pruned as well, as its malicious interfaces are identified. Therefore, the potential number of nodes, $n_{pr}(T, B)$, that can be pruned in a cycle from $\mathbb{T}$ using one PT B is as follows:

$$n_{pr}(\mathbb{T}, B) = 1 + \sum_{i=1}^{d_B - 1} \left(1 - tag(M_{B_{I_i}})\right) |T(B_{I_i})| \ . \tag{1}$$

While the PT B helps explore the *breadth* of the search space, the patrolling process proceeds hop-by-hop, quite similar to the prior works [13,15,40] in the literature. In the proposed solution, a second PT S inside $\mathbb{T}$ helps explore the *depth* of the subtree, so as to enable faster convergence. Let B_{I_j} be the interface of B under which S is located. Hence, $T(S) \subseteq T(B_{I_j}) \subset T(B)$. Let $S_{I_1}, S_{I_2}, \ldots, S_{I_{d_S}}$ be the d_S interfaces of S, where d_S is the degree of S. The Algorithm 2 implies the following Eq. (2).

$$n_{pr}(\mathbb{T}, B, S) = 1 + \sum_{i=1, i \neq j}^{d_B - 1} \left(1 - tag(M_{B_{I_i}})\right) |T(B_{I_i})| \tag{2a}$$

$$+ 1 + \sum_{k=1}^{d_S - 1} \left(1 - tag(M_{S_{I_k}} \| M_{B_{I_j}})\right) |T(S_{I_k})| \tag{2b}$$

$$+ \left(1 - tag(M_{B_{I_j}})\right) |T(B_{I_j}) - T(S)| \tag{2c}$$

$$. \tag{2}$$

Specifically regions (i),(ii) and (iii) of Algorithm 2 imply the terms (2a), (2b), and (2c) respectively of Eq. (2). The following inferences can be made from Eq. (2). **(i)** The term (2a) is not affected by the

choice of S; **(ii)** The term (2b) is high when S has a high $|T(S)|$ and degree d_S; **(iii)** The term (2c) is high when $T(S)$ contains all the attack-source routers and $|T(B_{I_j}) - T(S)|$ is high. This implies that choosing S is crucial. We next present the metrics that could be employed to make an effective choice of S.

4.2. Selection metrics

The objective of a selection metric is to help identify a PT S that can prune the maximum number of nodes from a given subtree-of-interest (henceforth referred to as $T(B)$[6]) and also quickly move closer to the sources of the attack. The proposed solution explores the following metrics to select a router as S: **(1)** *Distance*, to identify a router estimated to be the closest to the attacker; **(2)** *Centrality* to identify the most critical router in the network topology G in terms of its connectivity, traffic volume forwarded by it and/or the number of its interfaces; **(3)** *Subtree size* to identify a router having a large subtree underneath it in $T(B)$; and, **(4)** *Random* that identifies a random router in $T(B)$. The most interesting point which was empirically witnessed during simulation is that on the selection of S, if the attack-source router $x \in A$ is in $T(S)$, Net-Police moves closer to x, else a large part of the tree gets pruned as benign thereby reducing the search space. Both ways it turns out to be a win–win situation for Net-Police towards meeting its objective. We now analyze each of the aforementioned metrics in detail.

Distance (f_d). This metric estimates the distance in terms of the number of hops between the attacker and the victim. In this work, we use the *time-to-live*[7] (TTL) field in the IP header of the received malicious packets to get f_d. Since the attack traffic is voluminous, the victim observes the TTL values in the attack packets to identify the most frequent TTL values. The CA collects these TTL values from the victim to estimate f_d, and selects a router at a distance f_d from v in $T(B)$ at random as S. Such an S has the highest probability of being closest to the source of the attack.

Let S be connected to B through the interface B_{I_j}. It was found empirically that, S or one or many of its ingress interfaces are most likely to be malicious, resulting in assured pruning of some nodes in $T(B_{I_j})$ (refer terms (2b) and (2c) of Eq. (2)). The region $T(B_{I_j}) - T(S)$ is mostly benign as the attack sources are already found by S. Further, the closeness of S to the attack sources maximizes the term (2c) of Eq. (2). While an un-spoofed f_d can indeed help in estimation of the position of the attacker, it may not be reliable as the TTL values can be easily spoofed [42].

Centrality. This metric identifies the most critical node of G in $T(B)$ as S, such that S is highly likely to be on the attack path. Such a $T(S)$ is likely to contain most of the attackers, and $T(B_{I_j}) - T(S)$ can be pruned as a consequence (term (2c) of Eq. (2)). This scenario is highly likely on the Internet due to the presence of attackers in distributed clusters [31]. Additionally, such an S usually has a high $|T(S)|$ and a possibly high degree, resulting in a high value of the term (2b) of Eq. (2). In the event that S is not on the attack path, the entire $T(S)$ can be pruned. We now discuss four ways of estimating centrality.

Betweenness centrality (f_{c_b}) [43]. Betweenness centrality of a router node is the number of shortest paths in G that pass through it [43]. Intuitively, as network protocols route packets through shortest paths, a router with high betweenness centrality is more likely to be part of maximum such paths, including those carrying malicious traffic. Thus, this metric helps identify the most *connected* node of G in $T(B)$, which can be chosen as S. Such an S will have a high $|T(S)|$, but may not always have a high degree. To compute f_{c_b}, the CA executes the algorithm in [43] and requires the network topology and the shortest paths as inputs. Note that CA is aware of the topology G as well as the shortest paths (active routes) between the nodes in G (Refer Section 3).

Degree (f_{c_d}). Degree of a router is the number of interfaces it possesses. This metric helps choose a node S with a large number of interfaces, such that maximum number of benign interfaces can be pruned (Term (2b) of Eq. (2)), as well as detect attack sources in one of them if any.

Hybrid (f_{c_h}). This metric first identifies candidate nodes with a high value of f_{c_b} and further selects the node with the highest value of f_{c_d} among them as S. Such an S will result in a high value of the terms (2b) and (2c) of Eq. (2).

Traffic flow rate (f_{c_t}). The traffic flow rate is indicative of the cumulative real-time traffic volume forwarded by a router. Such a metric can potentially help identify a router which is likely to be on an attack path. The CA can retrieve this information from the routers using known techniques such as NetFlow [44,45]. Unlike f_{c_b}, f_{c_d} and f_{c_h}, which always choose the same central node in a given G, f_{c_t} incorporates the dynamic aspects of the attack to determine the central node in the network at a given instant of time. However, this metric may not always choose an S with a high $|T(S)|$ or a high in-degree.

Subtree size (f_s). This metric identifies an S such that a significant number of nodes can be pruned either in $T(S)$ or $T(B) - T(S)$. By choosing a reasonable sized $T(S)$ (say half of the size $|T(B)|$), this metric ensures that the number of benign nodes pruned is significant.

Random (f_r). This metric chooses an S at random in $T(B)$. The metric, though not recommended for in-field deployment, is used in this paper to serve as a baseline metric for comparison purposes, and establish a worst-case bound on the performance of Net-Police. Empirically, this metric does not yield encouraging results.

5. Secure packet tagging

In this section, we initially present the capabilities of an adversary that is aware of the presence of Net-Police. Further, we discuss how Net-Police handles each of these adversarial attempts. Specifically, we discuss the packet tagging mechanism employed to ensure reliable traceback of attack-source routers even in adversarial conditions. Finally, we analyze the impact of these security mechanisms on the performance of the routers.

5.1. Net-police adversary model

In Section 3, the DDoS adversary model comprising attackers who can cause a DDoS attack was presented. If all routers in the network are assumed to be *honest* (non-malicious), then Net-Police Algorithm 1 will identify the attack-source routers and its interfaces to which the attackers are connected. The next important question to ask is 'what if the routers are dishonest?'. In this section, we present an adversary model comprising attackers whose objective is to *mislead* the Net-Police algorithm. Such an adversary will be aware of the working of Net-Police, and can compromise *any* router in the network. The capability of such compromised routers is described as follows.

A1. When the router is designated as a PT, it
   (a) may not tag packets when asked to.
   (b) may tag packets with the correct RID, but incorrect interface information.
   (c) may tag packets with incorrect RID (illegal label).

A2. When the router is just forwarding the packets, it can tamper/sniff the labels in the packets passing through it. It can also deliberately tag packets using *forged* labels when it is not supposed to.

[6] Since the PT B is placed at the root of the subtree-of-interest.

[7] The TTL field in the IP header is known to help estimate the distance of the sender, as only few initial TTL values (For e.g. 128 in Windows and 64 in Linux) are used [41]. For e.g. If the TTL value in a received packet is 50, then the source of the packet is 14 hops away.

A3. Additionally, a router can also make spurious DDoS mitigation requests to the CA, which triggers unnecessary patrolling, and can affect the performance of the CA and the entire network. Such requests can also be used to learn critical information including IP addresses of different routers and the topology of the network.

The dishonest routers that can potentially mislead Net-Police through attacks A1 and A2 must be an intermediate router on the path between the attack-source router and the victim. The motivation behind such attempts is to conceal attacks originating from colluding sources or to frame other ISP routers. On the other hand, the router considered in attack A3 could be located anywhere in the network. We proceed to explain how these attacks are automatically taken care of in Net-Police.

5.2. Handling of attacks A1(a) and A1(b)

Net-Police starts patrolling without any prior information about the honesty/dishonesty of the routers. However, if any router behaves dishonestly, Net-Police will eventually identify such routers. Specifically, Algorithm 1 converges at the link connecting such a dishonest router to its honest parent in $T(v)$ as we show next. The Net-Police Algorithm 1 implies the following theorem.

Theorem 1. *For an attack on victim v, if Net-Police explores a subtree in $T(v)$, where attackers are not present, the subtree will eventually be marked as benign.*

However, in the presence of adversaries as described in Section 5.1 (A1(a) and A1(b)), there may be cases where a subtree in $T(v)$ could be falsely marked as benign even when there are attackers present in it. We first provide a concrete example of attack A1(a) using Fig. 5a. Here, the DDoS attackers in subtree labeled T_A send attack packets to victim $v(= R_k)$. Let us consider the case when router R_1 is dishonest, while $R_2, R_3 \ldots R_k$ are honest. Let $I_{j,j+1}$, $1 \leq j \leq k-1$ be the interface through which R_j is connected to R_{j+1}. When designated as a PT, R_1 deliberately does not tag packets from T_A. Consequently, v does not receive any labeled attack packets from R_1. As per Algorithm 2, Net-Police will prune $T(R_1)$(i.e., subtree rooted at R_1) from the search space, although attack packets traverse R_1. Effectively, Net-Police never re-examines a subtree marked benign within a given run. Nevertheless, Net-Police *will* detect a conflict at the dishonest router R_1 as we show next. During the patrol cycles, Net-Police infers that the subtrees $T_{j=2\ldots k-1}$ are benign, as none of the routers in $T_{j=2\ldots k-1}$ will tag packets belonging to the attack flow from T_A. But, any router $R_{j=2..k-1}$, when designated as a PT, will *honestly* tag these attack packets along with the interface $I_{j-1,j}$. Eventually, after a few patrol cycles, Net-Police detects a conflict at R_2 and its interface $I_{1,2}$ as v is still receiving labeled attack packets from R_2, while R_1 claims itself as benign. Thus, R1 is detected to be dishonest.

A similar argument can be made for the attack A1(b) (Refer Fig. 5a), wherein R_1 tags packets with incorrect interface I_b instead of I_a. Consequently, Net-Police continues to explore subtree T_B in Fig. 5a while marking R_1 and T_A as benign. From Theorem 1, T_B will also be marked as benign, and eventually Net-Police will detect a conflict at interface $I_{1,2}$ of R_2. Based on these arguments, we formulate the following theorem that shows the convergence of Net-Police either on the attack-source routers or the dishonest routers colluding with the attackers.

Theorem 2. *Given a path $(R_1, R_2, \ldots, R_k)$ from attack-source router R_1 to victim R_k, and let $I_{j,j+1}$, $1 \leq j \leq k-1$ be the interface through which R_j is connected to R_{j+1}. If $H = (R_t, I_{t,t+1}, R_{t+1}, \ldots R_{k-1}, I_{k-1,k}, R_k)$ be the maximal path of honest routers towards R_k, i.e. R_{t-1} is the closest dishonest router from R_k, Net-Police detects a conflict at R_t and its interface with R_{t-1}, namely $I_{t-1,t}$.*

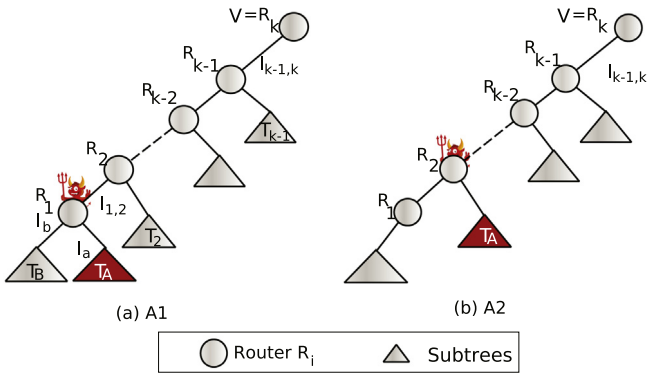

Fig. 5. Malicious behaviors A1 and A2 in Net-Police.

Proof. If a router R_p $(t \leq p \leq k-1)$ is enabled as a PT, the interface $I_{p-1,p}$ and the tree $T(I_{p-1,p})$ rooted at it will be marked as malicious for subsequent patrol cycles. No router R_q, $1 \leq q \leq t-1$, which becomes a PT can prune $I_{t-1,t}$. Consequently, CA detects a conflict at R_t and its interface $I_{t-1,t}$ to R_{t-1}, where R_{t-1} is the closest dishonest router to R_k. Note that R_{t-1} is sending malicious packets through $I_{t-1,t}$ to R_t while the subtree $T(R_{t-1})$ rooted at R_{t-1} is classified as benign. The CA detects the same. These and the argument preceding the statement of the theorem imply the theorem. □

The conflict detection step implied by Theorem 2 can be implemented as step 3-vii in Net-Police Algorithm 1, which should read as: In any subtree-of-interest, if a PT claims that its interfaces are malicious, while all descendant subtrees of the PT are explored and marked as benign, then either the PT is dishonest or its interfaces through which the attack packets are emanating are dishonest.

In a scenario when all other routers on the path are dishonest (i.e., $t = k$), Net-Police converges to the victim R_k, the only honest router, for filtering attack packets. However, in practice, a majority of ISPs would prefer to be honest so that the attack packets are filtered as early as possible, without crossing their networks.

5.3. Handling of attacks A1(c) and A2

An invalid label inserted either by a PT (attack A1(c)) or by any other intermediate router (attack A2) in between the attack-source router and the victim, can mislead Net-Police. Fig. 5b shows one such scenario, wherein the router R_2 (between R_1 and victim $v = R_k$), deliberately tags attack packets from DDoS attackers in the subtree labeled T_A with the label corresponding to R_1. As a result, R_2 can conceal attack sources in T_A as well as falsely implicate R_1 for the attack. To handle such attempts, Net-Police uses a secure packet tagging mechanism that ensures that a label received at the victim is reliable (i.e., valid, non-spoofed, and untampered). We first provide the details of this mechanism, followed by an analysis of the security it offers, and finally its performance implications.

5.3.1. Packet tagging mechanism

When a router is designated as a PT, it is supposed to tag packets, which are destined to the victim. It does so by inserting a label, which identifies the PT uniquely. To carry these labels, an additional field in the packet header is required. While the existing fields (e.g., fragmentation ID field) in the IP header could be used to carry this label [12,14, 16,17,46–50], re-using these fields can affect interoperability with the existing network protocols. For instance, overloading the fragmentation ID field could affect reassembly of fragmented packets. Even if a field that does not cause such interoperability issues can be found, the size of these fields will be limited to a few bits. This may make it easier for an intermediate router to forge these short-length labels.

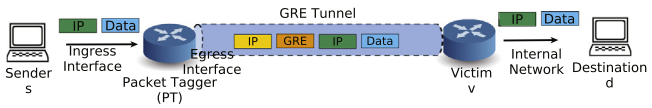

Fig. 6. GRE packet encapsulation and decapsulation.

Bits 0-3	4-11	12-13-15	16-23	24-31
C	K	S	RFU	T	Version	Protocol Type	
Checksum			Index in $\mathbb{DSI}$	Interface I
Key (RID)				
Sequence Number				

Fig. 7. GRE Header [52,53] used in Net-Police.

In this work, we use the Generic Routing Encapsulation[8] (GRE) protocol, which can carry labels that are up to 48 bits in length. The GRE protocol is also widely supported by both off-the-shelf routers and IPv6 networks [51], unlike many other tunneling protocols. Additionally, GRE also provides management fields such as *checksum* and *sequence number*, that can be leveraged to ensure the integrity of the labels. GRE works by establishing a tunnel between the PT and the victim. Note that the CA initiates the establishment of a GRE tunnel between the victim and the PT, and routes *all the packets destined to the victim* through this tunnel. The GRE tunnels provide a seamless carrier for the packets and does not affect the order of the packets. During the tunnel establishment, the CA generates a random identifier, namely, the RID, that identifies the PT uniquely at the victim. For this, the CA uses a pseudo random number generator[9] with a different seed for each victim. Additionally, the CA ensures that the RID generated for a victim does not match any other RID in the past 5 minutes for safety. This RID is communicated to both the PT and the victim. The PT uses the RID to create the label, while the victim uses it to verify the label it receives. The challenge is to transmit the label from the PT to the victim in a secure manner, thwarting adversarial attempts similar to A1(c) and A2.

All packets destined to the victim are *encapsulated* at the PT and *decapsulated* at the victim as illustrated in Fig. 6. The packet p received at the ingress interface of the PT comprises an IP header and data payload. Let the source of p be host h and the destination be a host d connected to the victim router v. The PT generates a GRE header as shown in Fig. 7, that in turn contains a unique label in addition to other fields. To prevent sniffing of labels by malicious adversaries en-route to the victim, the PT encrypts the GRE header before appending the same to the packet as shown in Fig. 6. In addition, the PT also adds a new IP header with the source as PT's address and the destination as v. The PTs address is a magic IP number provided by the CA, in order to hide the PT's identity en-route to the victim. Note that a valid address of a PT is not required at the victim, as the victim does not communicate back with the PT. Next, the PT transmits this new packet to v through a GRE tunnel specifically established by Net-Police for this purpose. The intermediate routers along the way parse the outer IP header alone, unaware of the GRE header. However, at the receiving end of the tunnel, the victim v, on receipt of a packet through the tunnel shall decrypt the GRE header and check the validity of the labels, and the malicious content in the packet[10] before deciding to forward the packet to the destination d or drop the same.

The APIs from the Linux kernel *ip_gre* and *ip_tunnel* modules are used for creating and using GRE tunnels. Normally, when GRE packets are transmitted/received, the routers do not encrypt/decrypt the GRE headers nor do they perform extensive validity checks. To ensure compatibility with the normal operations, the Net-Police implementation adds a new wrapper API *netp_gre* that invokes *ip_gre* and *ip_tunnel* in addition to performing encryption at the transmission end and decryption followed by validity checks at the receiving end. Any standard or customized encryption algorithm can be used for this purpose. The CA uses the *netp_gre* API for establishing GRE tunnels between the PT and the victim. We next describe how the GRE header and the label carried in the header are created at the PT and interpreted at the victim.

Inserting the secure label at the PT. The PT creates the label for a packet by concatenating a *dynamic* RID as mentioned earlier (that uniquely identifies the PT) and the ingress interface ID (I) through which the packet destined to the victim arrives. The need for a dynamic RID and its creation is explained as follows:

Encrypting the label field using a key which is shared between the PT and the victim ensures that no intermediate routers on the path can see the labels. However, a capable adversary can still learn the label that is being used, by employing statistical techniques on the labels observed over a period of time. Importantly, a PT (which implicitly have access to the encryption key) can learn the RIDs of other participating routers and can spoof RID of another PT (attacks A1(c) and A2 of Section 5.1). To counter such attacks, Net-Police uses a modified version of the *hopping labels* technique [54]. This technique mandates that the label identifying a PT is not *static* and that no two consecutive packets from the PT have the same label. To achieve this, instead of using a single RID for a PT, Net-Police uses a *dynamic array of random RIDs*, represented by $\mathbb{DSI}$ such that, any $RID \in \mathbb{DSI}$ can identify the PT. As $\mathbb{DSI}$ is an array, each of these RIDs is associated with a unique index wherein, it is stored in $\mathbb{DSI}$. As before, during tunnel creation, the CA computes the array $\mathbb{DSI}$ and communicates it to both the PT and the victim. For every packet, the PT selects a random RID from $\mathbb{DSI}$ to create the label. To ensure protection against statistical attacks, the size of $\mathbb{DSI}$ needs to be reasonably big. However, a large-sized $\mathbb{DSI}$ will affect the time it takes to check the validity of a given label at the victim, which will, in turn, affect the convergence time of Net-Police. To enable faster lookup at the victim, the PT also inserts the *index* of the currently used RID along with the label in the GRE header.

The *key* field and the *reserved for future use (RFU)* field of the GRE header (refer Fig. 7) is used by Net-Police for carrying the label and the index of the RID. The T bit (bit 12) which is part of RFU bits is set by the PT to indicate to the victim that the packet is tagged. Finally, the PT updates the checksum and sequence number fields in the GRE header, before encrypting the packet. Net-Police uses the symmetric AES-128 scheme [55] for encryption. Net-Police requires the key used for encryption and decryption to be established between the victim and PT, which is practical and feasible and hence widely assumed by existing works [56]. The CA provides this key to the PT and victim while establishing the tunnel. It is important to note that the presence of a dedicated secure communication channel between the CA and, the PTs and victim, secure the key distribution against eavesdropping (Refer Section 3). Table 2 summarizes the data configured by the CA at the PT and victim during tunnel creation. As evident, the configuration data required from the CA for *enabling* packet tagging in a router is minimal.

Interpreting the label at the victim. The packets received on established PT-victim GRE tunnels need to be interpreted by the victim. At the victim end, the tunnel end-point first decrypts the first 32 bytes of the received packet (using the key configured by the CA) to extract the GRE header. Next, the victim performs preliminary checks such as the validity of the checksum and sequence number field, and the presence of T bit, before extracting the label from the header. Subsequently, the

[8] Generic Routing Encapsulation creates a private point-to-point connection (a tunnel) between the two endpoints.

[9] The *ints()* method of the *Random* class in Java.

[10] Recall that the attack packets can be identified based on their signature (Section 3).

Table 2
Configuration data required for tagging.

Configuration data	Size (in bytes)
$\mathbb{DSI}$	$\lvert\mathbb{DSI}\rvert \times 4$
Symmetric Key	16
Tunnel identifier	4
Tunnel end point addresses	8
Remote end point IP address	4
Network to add route entry	4
Total	$36 + 4 \times \lvert\mathbb{DSI}\rvert$ bytes

victim checks the validity of the label by comparing the RID in the label against the expected RID. The victim uses the *index* extracted from the GRE header to *instantly* look-up the expected RID from the $\mathbb{DSI}$. All invalid labels (either tampered/spoofed) are thus ignored by the victim. The victim stores only the valid labels found in attack packets for reporting to the CA, thereby ensuring that the CA processes only valid (non-tampered and non-spoofed) labels. This results in the CA pruning the subtree rooted at the dishonest PT as benign. However, as proved in Theorem 2, Algorithm 1 will identify such a PT.

5.3.2. Security analysis of packet tagging

At a malicious intermediate router between the PT and the victim, any attempt to tamper/sniff/forge/remove labels require deep packet inspection to identify the presence of labels in the packets that it forwards. While this is computationally expensive, we analyze the security of packet tagging in the presence of a powerful adversary having such capabilities.

Any attempt to forge labels (as in Fig. 5b) can succeed only if the malicious router is able to deduce the set of valid labels used by a PT before it. Since the labels are encrypted, compromising the encryption keys becomes necessary in order to read the labels. The probability of compromising the key is $\frac{1}{2^{128}}$ [55], even in the case when the adversary has access to plain-text and cipher-text pairs. Even if the adversary successfully compromises the key and learns the labels, the labels can only be used within the same patrol cycle. The time period of a patrol cycle is estimated to be less than 1.2s (Refer Section 6.1.1). Since the labels are generated at random for each patrol cycle, the malicious router has to guess the 32-bit RID used by the target PT. Thus, the probability p_h that the label is valid in a patrol cycle is $\frac{\lvert\mathbb{DSI}\rvert}{2^{32}}$. Therefore, the probability of compromising the key and further inserting a valid label is $p_e \times p_h = \frac{\lvert\mathbb{DSI}\rvert}{2^{160}}$. An entropy of ≈ 160 bits is sufficient to ensure security against even a powerful adversary [57].

A malicious intermediate router may deliberately remove the labels (stripping off the GRE header) from a tagged packet before forwarding the packet. While such an attack is possible in the current form of Net-Police, it can address such adversaries by encrypting the inner IP header along with the GRE header during the encapsulation at the PT. Consequently, even if the adversary strips the GRE header, the inner IP packet is unsuitable for forwarding, and hence dropped, unless the encryption key is also compromised. Such attempts in turn benefits the victim by thwarting the DDoS. Addressing such an adversary adds an overhead of encrypting one 128-bit block at the packet tagger.

Finally, a malicious router may deliberately drop labeled packets passing through it. While such an attempt hinders traceback, it actually benefits the victim by thwarting the DDoS attack. However, the malicious router may also end up dropping labeled packets that are legitimate. This challenge is not specific to Net-Police, and there are known methods to deal with such attacks [58].

5.3.3. Secure PT performance implications — An analytical study

GRE encapsulations can potentially impact the throughput of downstream routers on the path to the victim. In particular, the increase in packet size can affect the packets per second (PPS) that a router can transmit over a link.

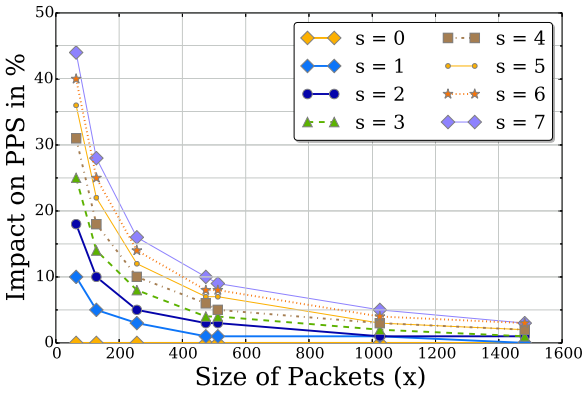

Fig. 8. Impact on PPS at R, in presence of multiple simultaneous PTs.

To quantify the impact, we consider a router R on the path with an outgoing link of bandwidth capacity of C bits-per-second (bps). Let n be the number of packets that R can transmit (forward) per second on the link. Assuming all packets that R receives are of size x bytes, n is given as

$$n = \frac{C}{8 \times x} \quad . \tag{3}$$

Let s denote the number of PTs that the packets have traversed through before it reached R. Every PT on the path, adds 36 bytes to each packet that it tags; the GRE header of 16 bytes (Fig. 7) and the outer IP header[11] of 20 bytes. Hence, labeled packets are of size $x+s\times36$ bytes. Let l denote the fraction of the total number of packets that arrive at R that are labeled. Hence, the number of packets that R can transmit per second on the link is given by

$$n' = \frac{C}{8 \times (x + l \times s \times 36)} \quad . \tag{4}$$

Fig. 8 plots the impact on PPS when 20% of packets are labeled, on a link of capacity 10 Mbps, for varying packet sizes. From the plot, we can see that a higher s degrades the performance of R, as expected. Further, R is most affected when processing smaller sized packets, as the number of such packets processed by R per second is high. For $s = 2$, the impact is less than 4% for packets of size > 256 bytes, and 18% for packets of size 64 bytes. However, this overhead is restricted to a short time of $\lesssim 350ms$, when packet tagging is enabled. Recall that designating more than one PT simultaneously, in a subtree-of-interest is advantageous to quicken the convergence (Refer Section 4.1). Hence, we can conclude that when $s = 2$, the overheads of packet tagging on R is within acceptable limits.

The fraction l of labeled packets in the traffic that R has to forward, depends on the distance of R from the victim. While l is low when R is far from the victim, it can be considerably higher when it is closest to the victim. Fig. 9 plots the impact of varying the fraction of labeled packets l on PPS when $s = 2$. For higher packet sizes, the effect of l on PPS is negligible. However, for smaller sized packets, the effect is significant. For instance, for 64 byte packets, a 10% increase in l degrades the PPS by 10%. For average packet sizes observed on the Internet i.e. ≈ 500 bytes [59], we can see that the impact is less than 10% even if l is close to 80%.

5.4. Handling A3

An adversary posing as a "victim" can make spurious Net-Police service requests, either to cause a DoS at the CA or to learn critical information about the ISPs. These spurious service requests can potentially *overload* the CA, resulting in the CA being inaccessible to other

[11] Assuming that IPv4 is used as the delivery protocol, the minimum size of the IPv4 header is 20 bytes. In case of IPv6, the size of the header is 40 bytes.

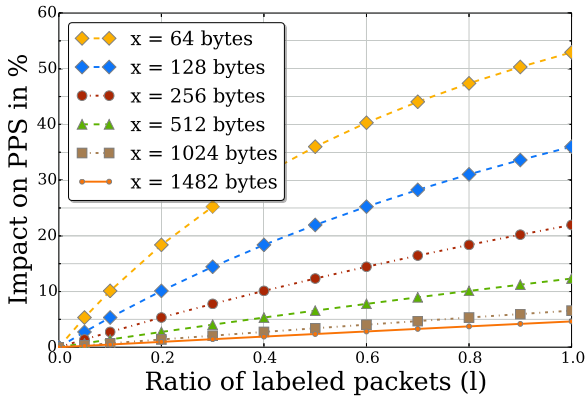

Fig. 9. Impact on PPS at R, when $s = 2$.

legitimate victims. However, in a pay-and-use economic model, the misbehaving victim will end up paying for each spurious request. We argue that the monetary cost involved in the attack will discourage such misbehaviors. In addition, the CA could include simple mechanisms such as collecting traffic flow rates from intermediate routers in order to verify the authenticity of such requests. Alternatively, CA could ask the victim to produce evidence that substantiates its service request before triggering Net-Police. Defining such evidence and the corresponding verification mechanisms are important considerations for future work.

A misbehaving victim could also use spurious service requests to learn critical information such as IP addresses and network topology. For instance, the IP address of a PT could be learned every time a tunnel is established. This information could be misused to initiate attacks at the respective ISPs [60]. However, we argue that by designating a small number of dedicated IP addresses to be used for the tunnel end-points, the extent of such attacks can be limited. Further, the misbehaving victim may attempt to infer the topology of the network using the received labels and the corresponding IP addresses of PTs. While this is still possible, there are alternative mechanisms, which are both cheaper and readily available, to infer such information [61].

6. Implementation and evaluation

In this section, we discuss the implementation details and evaluation of Net-Police. Our implementation consists of two core components, namely, the Net-Police control logic that exists at the central authority (CA) and the control/data plane at the routers. While any protocol could be used for the communication channel between the CA and the routers we choose OpenFlow for the ease of implementation. The CA functions were implemented as Java module. In our setup, we instantiated the CA on an Intel Xeon E5620 Server. On the other hand, the setup of interconnected routers were realized in the following three ways for evaluation: **(E1)** a custom-made *simulation* model capable of supporting at least 63,463 routers, for a comprehensive evaluation of Net-Police; **(E2)** a *physical test-bed* of 30 routers built using off-the-shelf hardware, for evaluating practical feasibility; and, **(E3)** a *physical packet tagger* (PT) using an Intel i7 machine for evaluating the performance. We use our experimental setup for answering the following questions:

- Q1 Which *selection metric* of Net-Police helps identify (i.e., traceback to) the attack source quickly?
- Q2 How does Net-Police perform compared to prior traceback-based schemes on real-world networks?
- Q3 Can Net-Police respond to real-world attack scenarios, such as, the Mirai attack in 2016 [1]?
- Q4 Does Net-Police really benefit both the victim and the ISPs? How does it perform compared to existing DDoS defense solutions?
- Q5 Does the position and number of attackers affect the time required to traceback?
- Q6 Does the position of a victim, relative to a given set of attackers (bots), affect the traceback?
- Q7 What is the estimated number of PTs that are enabled per service request (from the time of request by the victim to the time of convergence)?
- Q8 Can the Net-Police data-plane be implemented on an off-the-shelf router hardware?
- Q9 What is the performance overhead at a router due to packet tagging? For instance, what is the number of packets that the router can process per second?

The questions Q1 to Q7 are answered using our simulation environment (E1), whereas the questions Q8 and Q9 are answered using the physical testbed (E2) and the Intel-i7 physical packet tagger (E3) respectively. We now discuss the details of our experimental setup and evaluations.

6.1. E1: Simulation environment

In this section, we first describe our simulation setup. Following this, we perform evaluations with an aim to answer Q1 to Q7.

6.1.1. Experimental setup

Answering the aforementioned questions about the effectiveness of Net-Police requires a setup that can support a wide range of topological and behavioral variations (of both attack and legitimate nodes). For this, our custom simulation framework written in Java supports the following: **(1)** realization of large network topologies; **(2)** random placement of the victim node on these topologies; **(3)** variations in the number and position of attackers on these topologies; **(4)** replaying both real-world and synthetic traffic traces; and, **(5)** generating malicious traffic from the network nodes using known attack signatures [62].

Topology: We evaluate Net-Police on the AS-level topology [20] of size 63,463 nodes, which may not be an accurate picture of the Internet [37]. However, today, we are limited by the availability of a clearer picture of complete AS-level paths in the Internet. Additionally, for heterogeneity and exploratory purposes, we also use 256 real-world topologies from the Internet Topology Zoo (ITZ) [21]. These topologies have a size ranging from 23 to 753 nodes. Specifically, we use the ITZ topologies to get insights on the performance of Net-Police on different topological structures.

Placement and number of attackers. For the smaller graphs from the ITZ library, the number of attack-source routers was assumed to be between 1% and 10% of the total number of routers. These routers are placed randomly in the network. While Net-Police can traceback any number of attack-source routers in the network, the assumption of 10% of the total routers being attack sources is sufficient to cover even the worst-case real world scenarios. For the larger AS-level Internet graph, we use the following insights from prior studies [15,31] to decide the number and placement of the attackers: **(1)** During an attack, most of the attack-source routers are known to be clustered within a few countries, typically < 7; and, **(2)** Very few routes in the Internet span more than 32 hops. Therefore, we realize 7 such attack clusters for each experiment based on the first insight. An attack cluster is formed by selecting a random router p, and further designating 1% to 10% of routers within a distance of 5 hops from p as attack-source routers. Based on the second insight, we place the attackers within a distance of 32 hops from the victim.

Estimation of Net-Police Patrol Cycle Time. Our simulation framework reports the number of patrol cycles needed to traceback all the attack-source routers. However, we can also estimate the actual time taken by Net-Police for the traceback. We estimate the actual time taken by multiplying the number of patrol cycles with the time taken

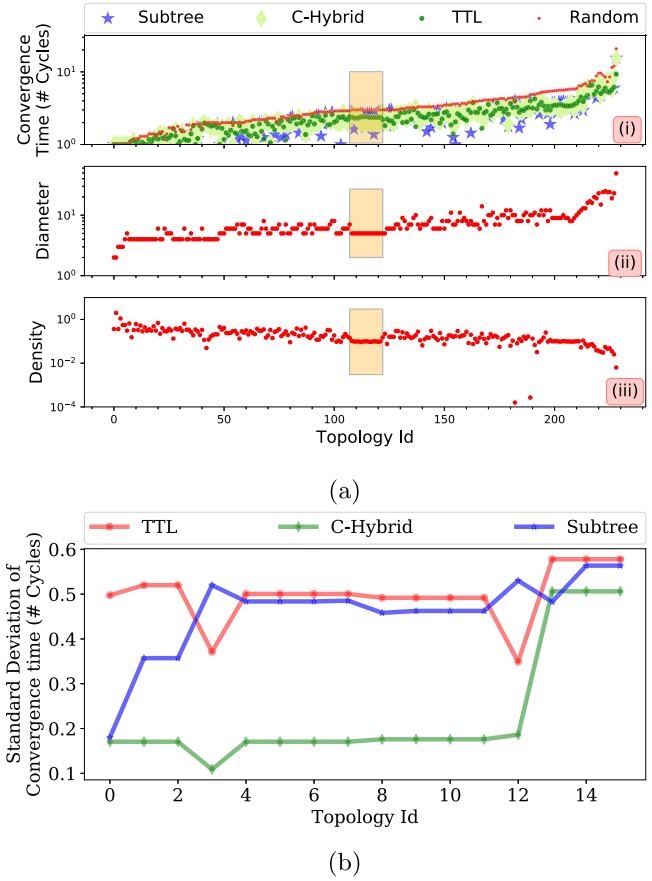

Fig. 10. Analysis on Selection Metrics. **(a)** (i) Convergence time (number of cycles) observed in different topologies; (ii) and (iii) Trends in Diameter and Density of the topologies; **(b)** Consistency of different metrics for a subset of topologies selected from the highlighted box in (a), that have the same diameter and density.

per patrol cycle. We note that the major amount of time spent per patrol cycle (step 3 of Algorithm 1) is due to the communication delays in three steps, namely, 3[ii], 3[iii], and, 3[iv]. In smaller-sized ITZ topologies [21], since the routers are less than 5000 km apart, the communication delay in each step will be around 100 ms [39], thereby resulting in a total patrol cycle time that is three times that, i.e., 300 ms. For the larger AS-level topology, as the routers can be more than 10,000 km apart, communication delay in each step is estimated to be around 300 ms [39]. Hence, we fix the patrol cycle time for AS-level topology to be 900 ms.

6.1.2. Results

We now present the experiments conducted and discuss the results.

Q1. The best selection metric. Different selection metrics, namely, distance (f_d); centrality-based (betweenness centrality (f_{c_b}), degree (f_{c_d}), hybrid (f_{c_h})), and traffic flow rate (f_{c_t})); subtree size (f_s); and, random (f_r) designate different routers as PT in each patrol cycle, thereby resulting in different convergence times. At the same time, the convergence time can vary based on the network topology, and the location of the attackers in the network. We define a metric as the most suitable for Net-Police if it can consistently yield the least convergence time, irrespective of the network topology or the attack scenario. To identify such a metric, we run Net-Police against the topologies in the ITZ library as well as the AS-level Internet topology. We run experiments for each selection metric in 1000 different combinations of attacker-victim configurations for statistical significance.

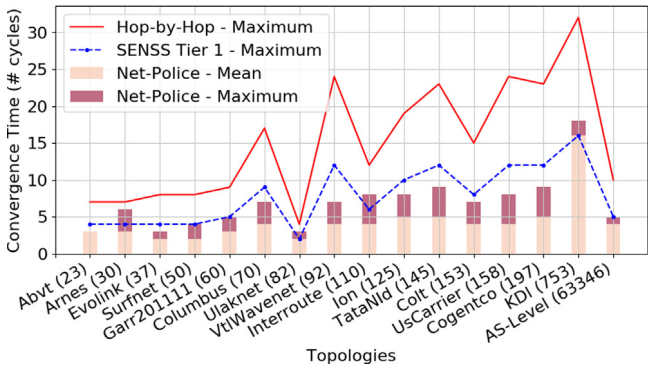

Fig. 11. Comparison with prior traceback schemes in different networks. In each topology, 10% of nodes are attack-sources. The size of the network is indicated along with the name.

Fig. 10(a)(i) plots the average convergence time observed with each metric in the 257 topologies ((in log scale). We only plot the convergence time for the top three of the seven metrics: f_d, f_s, and f_{c_h}, and the f_r metric for reference. From the figure, we can see that unlike a random choice of PTs, an intelligent selection based on a metric helps Net-Police converge on the attackers faster. In most topologies, f_{c_h} metric yielded the least convergence time, while in few cases f_s metric outperformed f_{c_h} metric. Intuitively, f_{c_h} (a combination of both f_{c_b} and f_{c_d}) helps identify a node that is both critical (based on betweenness centrality) and well-connected (based on degree centrality) as a PT, thereby assisting Net-Police to explore large chunks of the search space in each cycle. Interestingly, in our experiments, Net-Police took longer to converge using f_{c_t}, as the cumulative traffic flow rate at a node is potentially not conclusive of the node being on the path towards the victim. Further, to analyze if topology impacts the efficiency of a metric, we also plot two characteristics of the topology graph, namely, its diameter and density in Fig. 10(a)(ii)&(iii). We can observe that the convergence time is directly proportional to the diameter of the topology, while being inversely proportional to the density of the topology. Intuitively, in denser graphs, Net-Police is more efficient since it can prune large portions of the graph per patrol cycle.

To assess which metric is better, we analyze a subset of topologies from Fig. 10(a) that have the same diameter and density, marked by a highlighted box. For these network topologies, we compare the consistency of each metric across 1000 different attacker-victim configurations. Fig. 10(b) plots the standard deviation of the convergence time for each metric for the 1000 configurations. We can observe that f_{c_h} is the most stable metric. *Henceforth, for further analyses, we configure Net-Police to use the C-Hybrid metric f_{c_h}.*

Q2. Comparison with existing traceback schemes. We compare the quickness of Net-Police (with f_{c_h}) against light-weight traceback-based solutions [13,15,18] that follow the *hop-by-hop* approach, but use different mechanisms to ascertain the sources of attack. Although in all the three schemes, a hop-by-hop approach is used, the AS at which the traceback stops is different: SENSS [18] stops at a Tier-1 AS, whereas [13,15] stop at the attack source. For these approaches, in Fig. 11, we plot the maximum time taken (in terms of the number of cycles/hops) when the attacker is the farthest from the victim on 16 different topologies. The former is marked with a dashed blue line in the figure, and the latter is marked with a solid red line. For Net-Police, in the figure, we also plot both the mean and maximum convergence time under different attack scenarios. In comparison to prior solutions, Net-Police is able to converge faster irrespective of the number or position of the attack-source routers. For instance, Net-Police converged in ≤ 9 cycles ($2.7s$) in Cogentco network, and in four cycles ($3.6s$) in AS-level graph, even in the worst-case conditions. Even though the gains of Net-Police are only in the order of a few seconds in some cases, the benefits it provides are immense. This is because the volume of attack

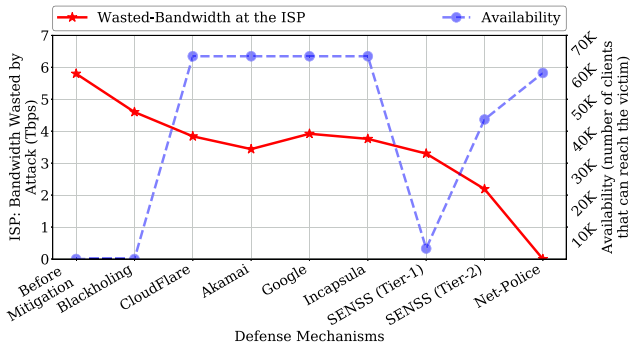

Fig. 12. Comparison of DDoS defense solutions against the evaluation metrics..

traffic on the ISP links is in the order of *Tbps* and mitigating the attack a second earlier saves terabits of attack data from transiting [6].

Q3 and Q4. Real-world attacks and comparison with state-of-the-art. To evaluate if Net-Police can respond to real-world attack scenarios, we reproduce the 2016 Mirai attack against Dyn [1] on our Internet AS topology simulation. We designate 15 of the ASes as attack sources as per the findings of prior works that analyzed the Mirai attack [63,64]. We assume that a sum total of 1.2 Tbps attack traffic is produced by these attack sources towards Dyn (AS11049). We observed that Net-Police (with f_{c_h}) was able to identify the 15 ASes containing the attack-source routers in four patrol cycles ($\approx 3.6s$). The filters employed at these AS'es relieved Dyn from the attack while ensuring the Dyn IP was reachable to 92% of its client-ASes. Note that 8% of the client-ASes shared a route through the attack-source AS. Furthermore, it also saved bandwidth by not allowing attack traffic to transit the ISP links.

We now compare the performance of Net-Police against the state-of-the-art techniques for DDoS mitigation in terms of two parameters: service availability to client AS'es and the cumulative bandwidth wasted at the ISPs. We present the results that correspond to the Mirai attack against Dyn. Fig. 12 plots the aforementioned two parameters for Net-Police, SENSS, the cloud-based solutions (Incapsula, Google, Akamai, CloudFlare), and Blackholing. As a reference, we plot the case where no mitigation scheme was employed. From the figure, we can observe that, before mitigation, the ISPs expend roughly 5.8 Tbps of bandwidth on its links due to the attack. In the case of Blackholing, the ISP links still expend 4.6 Tbps of bandwidth to carry the attack traffic until the first-hop ISP where Blackholing is employed. Similarly, with the cloud-based solutions, ISPs expend around 3.76 Tbps of bandwidth while carrying the attack traffic to the closest scrubbing centers. On the other hand, with SENSS filters at Tier-1 and Tier-2, the ISPs expend only 3.39 Tbps and 2.19 Tbps of bandwidth respectively. In contrast, in Net-Police, with filters deployed closer to the sources, the wasteful consumption of ISP links is $< 1\%$. In terms of availability, Blackholing performs the worst since most legitimate clients cannot reach the victim. In contrast, cloud-based solutions perform the best by ensuring that the legitimate traffic is forwarded to the victim. In the case of SENSS, when the filtering is done at Tier-1, only 3286 number of clients could reach the victim. However, when the filtering is done at Tier-2, the availability improves to 43,669 number of clients. Net-Police is comparable to state-of-the-art works by providing an availability of 92%. Some legitimate clients could not reach the victim since the only path to the victim was through the attack source AS. We would like to reiterate that the cloud-based solutions also have the drawback of privacy issues as compared to Blackholing, SENSS, and Net-Police.

Partial Deployment. We evaluate the benefits offered by Net-Police when only a certain percentage of ISPs on the Internet participate. To that end, we consider multiple deployment scenarios on the AS-level topology, as explained next. We vary the number of ASes participating in Net-Police, and for each percentage of participation, we consider

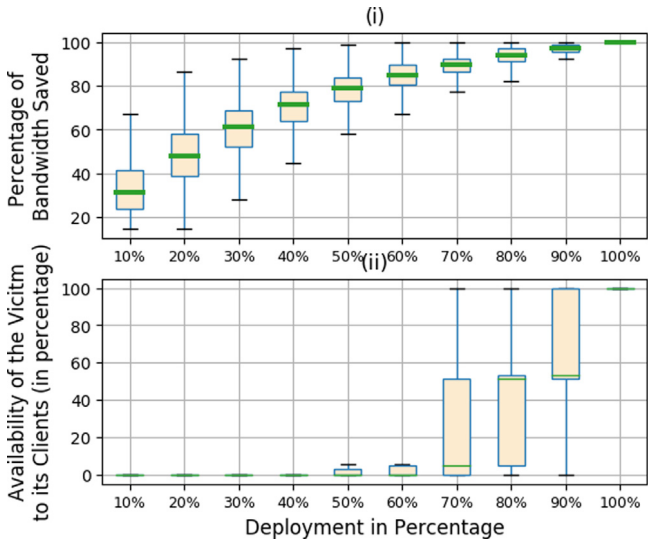

Fig. 13. Benefits of Net-Police under Partial Deployment. **(i)** Total percentage of bandwidth saved at the ISP links from being consumed by the attack; and **(ii)** Availability of the victim to its clients. The green line inside the colored boxes indicates the median value observed. The colored boxes and the blue whiskers indicate the ranges of values observed for 50% (5 million) and 99% (9.9 million) of the deployment configurations, respectively.

10 million different deployment combinations (actual positions of the ASes) on the Internet. Fig. 13 presents the results corresponding to the mitigation of the Mirai attack against Dyn in these scenarios. We use two parameters for evaluation: the percentage of total ISP bandwidth saved from being consumed by the attack[12] (Refer Fig. 13-(i), and the percentage of clients that can reach the victim (Refer Fig. 13-(ii). When all ASes participate (100%), Net-Police provides the maximum benefits in terms of total bandwidth savings and availability of the victim. However, when participation is lower, the benefits offered by Net-Police reduces. We note that ISPs can save about 50% of bandwidth, even with 20% deployment. However, for seeing any benefit in terms of availability, Net-Police needs to be deployed on 70% of the AS'es. Specifically, in lower deployment scenarios, although Net-Police stops the DDoS flows on the attack path, the victim gets cut-off from the rest of the Internet, even with a single filter rule closer to the victim. We acknowledge this as a drawback of our solution, especially when considering incremental deployment and hope to address this in future work. At the same time, we also expect widespread adoption of Net-Police, as the efforts involved in instrumenting the ISP, to support and participate in Net-Police is very minimal (as discussed in Section 7).

Q5. Impact of number and position of Attackers. It is natural for an adversary to place a large number of attacker bots strategically so that the DDoS attack can sustain longer. To evaluate such scenarios, we consider 1000 random positions of attack-source routers in the Cogentco network, and plot the CDF of the observed convergence time in Fig. 14(a). We can observe that irrespective of the number of attack-source routers or their relative positions, Net-Police detects all the attack-source routers within 9 cycles (2.7secs). These plots also highlight that there were some positions at which the attacking bots were quickly discovered. At the same time, there were a few other positions where Net-Police takes longer to discover them. For instance, when the number of attack-sources were assumed to be less, attackers were detected as quickly as 3 cycles ($900ms$) in 19% of the cases, whereas it required > 9 cycles ($> 2.7s$) in 2% of the cases. We conclude that even a clever adversary can only delay the detection of the attack bots but cannot avoid it.

[12] Note that a total for 5.8 Tbps of bandwidth was wasted on the ISP links during the Mirai attack.

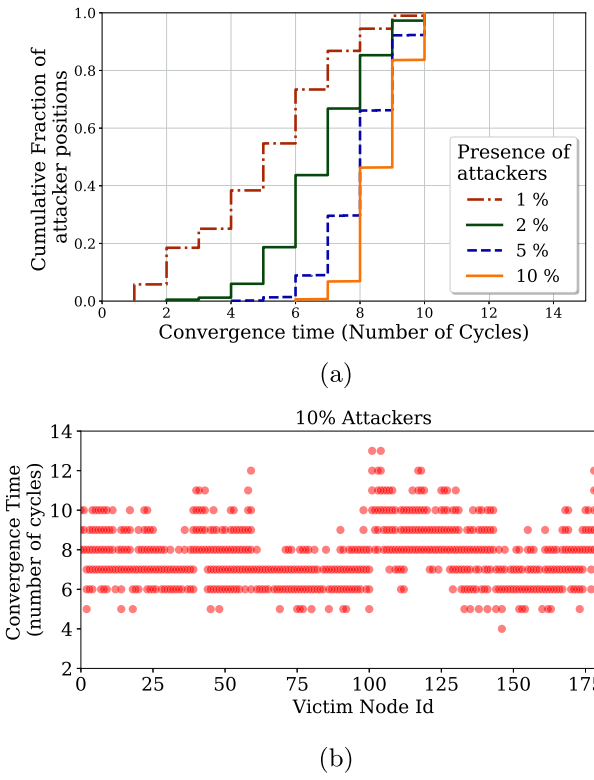

Fig. 14. Analysis on attacker and victim position. **(a)** The CDF of convergence time for different attacker configurations; **(b)** Convergence time for different victim positions.

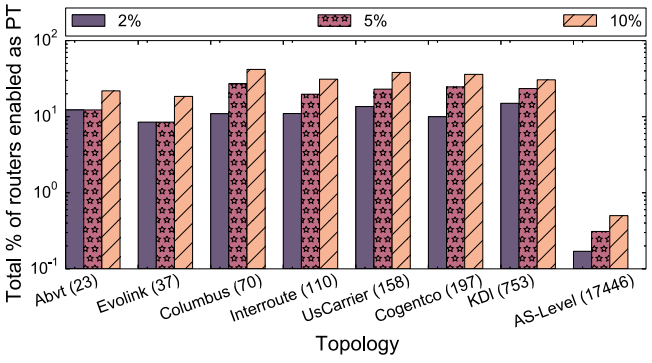

Fig. 15. Aggregate number of PTs required per Net-Police service request for different networks. The size of the network is indicated along with the name.

Q6. Impact of position of the victim. We now evaluate if some nodes in the network are more vulnerable to DDoS attacks due to their position. We consider the Cogentco network of size 197 routers, in which we designate 10% of the routers to be attack sources. We evaluate the impact when each of the 197 nodes in the topology is designated as a victim. For each victim configuration, we run the experiment 50 times with different attacker configuration. Fig. 14(b) plots the convergence time for each victim position. Most positions exhibit variable convergence times based on the attacker configurations. In such cases, Net-Police can protect the victim in as low as 5 cycles to as high as 13 cycles. Interestingly, we can also observe that there are some (30) victim positions which can be protected quickly by Net-Police. Such positions are coveted and it makes sense to use such AS'es for deploying critical services.

Q7. Number of PTs per service request. The total number of routers that are enabled as PT for each service request is a measure of the overhead of Net-Police. Fig. 15 plots the percentage of PTs enabled (in log scale) in different network topologies for three different percentages of attack-source routers (2%,5%, and 10%). As expected, tracing higher percentage of attack-source routers requires more number of routers to be enabled as a PT in every topology. Note that the percentage of routers that need to be enabled as PT are significantly higher in the ITZ topologies since we assume worst-case random placement of attackers. In contrast, for the AS-level topology, we followed a more realistic distribution of attack sources, resulting in modest percentage ($< 1\%$) of routers being used as PTs.

6.2. E2: Physical test-bed

The objective of this setup is to answer the question Q8, related to the practicality of an off-the-shelf implementation.

6.2.1. Experimental setup

To evaluate the practical feasibility of Net-Police, we build a *close-to-real-world prototype* of the data plane using off-the-shelf hardware. Fig. 16(a) illustrates a test-bed created using 30 Intel Galileo boards [65]. Each Galileo board in the test-bed runs Linux v3.14.28 and is configured as a router. It can also tag packets when enabled by the CA. The packet tagging mechanism is implemented as a Linux kernel module, by instrumenting the vanilla *ip_gre*, *gre* and *ip_tunnel* kernel modules to support Net-Police specific features. Since the Galileo board is equipped with only one Ethernet port, all the boards in the test-bed are physically connected in a bus topology. Nevertheless, any network topology can be simulated on the test-bed as explained next. We use the IP addresses assigned to each board to manage the test-bed.

Configuring the Topology. A remote management engine running at the CA configures the test-bed for any given topology of ≤ 30 routers. The engine first determines the number of boards required, and initializes the required services like IP forwarding on each of them. Next, on each board, the engine simulates the required number of virtual interfaces as per the topology and assigns virtual IP addresses for these interfaces. These virtual interfaces form the data-plane. Finally, the engine computes the routes for the topology and configures the static routes on each board for packet forwarding. Consequently, a packet from any source to any destination in the test-bed follows the route through these virtual interfaces. In the test-bed, the engine supports random placement of victim and attack-source routers; and can also replay benign and malicious traffic. The test-bed can thus be used for studying on-field feasibility and validation of network services such as the proposed solution.

6.2.2. Results

To demonstrate the effectiveness of Net-Police, we configure the test-bed as per the AboveNet topology [21]. Random attack-source routers in the test-bed send malicious traffic to a victim (chosen in the network), which invokes the Net-Police service when the incoming traffic at its end crosses a threshold. Fig. 16(b) plots the number of attack packets observed at the victim with and without using the Net-Police service. The plot shows that Net-Police is able to quickly detect the sources of attack and filter the malicious traffic at the sources.

6.3. E3: Physical packet tagger

The objective of this setup is to answer the question Q9, related to the overheads of Net-Police.

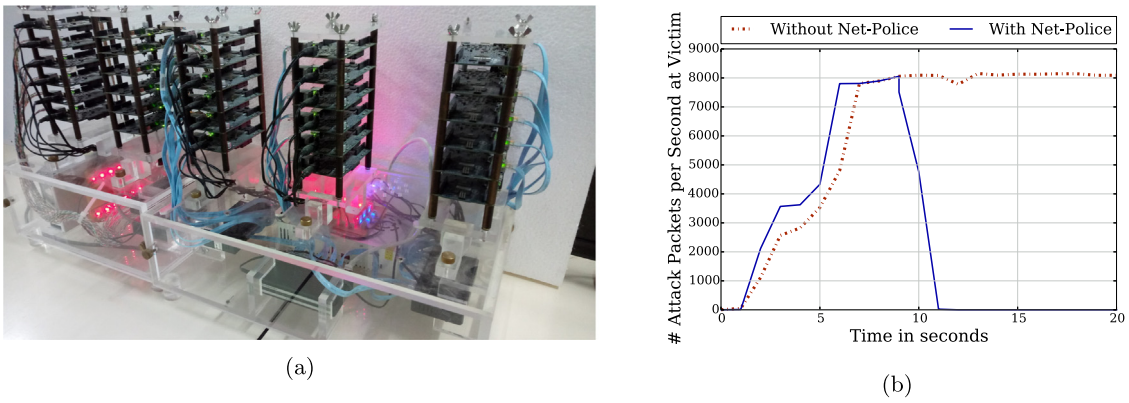

Fig. 16. (a) Physical Test-bed; (b) The number of malicious packets per second at the victim with and without Net-Police.

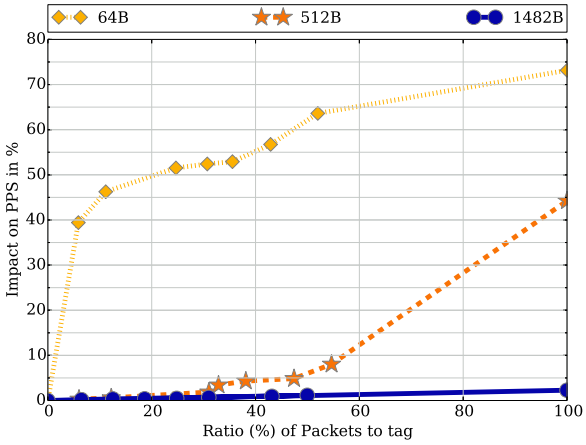

Fig. 17. The percentage impact on PPS throughput of a PT.

6.3.1. Experimental setup

While the test-bed showcases the practical feasibility of Net-Police on a topology of 30 routers, it cannot be directly used to evaluate the performance overheads since most real-world routers are much powerful than Galileo boards. To assess the overheads due to tagging (which includes encryption), we use an Intel i7 machine as PT, which is as powerful as the routers available at the ISPs. The machine (with Intel i7-7700 CPU, 8 cores and 64 GB RAM) runs Linux v4.17.13, and is equipped with a dual port Intel 82599EB X520-DA2 10GbE NIC. The `quagga` service is used to enable the machine as a router [66]. As in Section 6.2.1, kernel modules are instrumented to implement packet tagging. Two other Intel i7 machines equipped with Intel 10GbE NICs serve as the traffic generator and the victim respectively. The traffic generator is the source sending traffic at 10 Gbps to the victim via the PT, which tags incoming packets before forwarding them. We used techniques from prior works [67] to increase parallelism in the networking stack on the multi-core i7 systems. We now compare the throughput of the PT in terms of packets per second (PPS) that it can forward, with and without packet tagging enabled at it. In the incoming traffic of 10Gbps, we vary both the size of the packets and the percentage of packets destined to the victim. Accordingly, the percentage of packets that the PT has to tag varies.

6.3.2. Results

Fig. 17 plots the impact of tagging in terms of degradation in PPS throughput of the PT, against the percentage of packets that it tagged. From the plot, we can see that for traffic containing large-sized packets, the impact is $\approx 3\%$, even when the PT tags 100% of incoming traffic. However, for smaller sized packets (64 bytes), the impact is significant. This is because the number of packets processed by the PT per second is high. However, for average packet sizes observed in the Internet ($\approx$ 500 bytes), the impact is $< 5\%$, even when the PT tags $\approx 50\%$ of the traffic. Note that Fig. 17 is in line with the analytical analysis of transmission delay presented in Section 5.3.3. While the analytical analysis considered only the transmission delay, in this empirical analysis, we account for both the computation time required to tag the packets as well as the transmission delay at the PT.

Additionally, the encryption involved in the packet tagging can delay the traffic towards the victim. The time required for an AES-128 encryption on an i7 machine was found to be 0.4 μs. This resulted in an overall delay of $\approx 2\mu s$ (two encryptions + two decryptions) for a given network packet, which we think is tolerable on the Internet.

Comparison of overheads. Table 3 compares the overheads at the ISPs in the case of Net-Police, SENSS, and MOD. We measure the overhead per service request at a router in each solution in terms of CPU usage and impact on PPS throughput. We measured the CPU usage and PPS throughput at the router when it forwards 10Gbps of traffic of which 50% is attack traffic. Net-Police incurred a 2% CPU usage, whereas SENSS and MOD incurred a CPU usage of 1% and 20% respectively. In terms of PPS, Net-Police resulted in a 5% degradation while SENSS and MOD did not show any visible degradation. This is primarily because SENSS relies on traffic summary buffers that are not on the packet forwarding path at the router, whereas MOD in its current form does not deploy any security mechanism to secure the labels. However, it is important to note that Net-Police incurs these overheads only when a router is enabled as a PT ($\approx$ 350 ms) while both SENSS and MOD rely on the router's participation even under non-attack conditions.

7. Discussion

In this section, we first discuss the deployment requirements of Net-Police at a participant ISP/reflector. We also analyze if an adversary can delay convergence time, or evade Net-Police by moving to a benign subtree.

Deployment Requirements at an ISP/Reflector. Net-Police can be easily integrated into the existing defense mechanisms at the ISPs/reflectors and does not require any expensive equipment or services. In essence, any participating AS needs to be instrumented in the following ways: **(1)** When the DDoS attack is detected, inform Net-Police with the signature of the packet. Note that most ISP ASes/reflectors already have in-house DDoS detection mechanism such as the threshold violation by traffic flows [68]; **(2)** Ability to create GRE tunnels. Note that almost all commercial routers support the GRE protocol [52]; and, **(3)** Process GRE packets. Note that the corresponding overheads are very low as we show in our evaluation in Section 6.3.2. At the same time, ISPs also have strong incentives to deploy Net-Police to attract more customers, specifically the ones who demand high availability (e.g., e-commerce services), where even

Table 3
Comparison of overheads at the ISPs.

Overheads		Net-Police	SENSS [18]	MOD [17]
Impact at router	CPU usage	2% once for 350 ms only during the attack	1% for 10s at regular intervals	20% always
	PPS throughput	<5% once for 350 ms only during the attack.	NIL	NIL[a]

[a]Without any security measures employed

a minute of downtime can incur a significant loss [6]. While the ISPs can decide on the number of its ASes that participate in Net-Police, higher participation results in better protection against attacks. We believe that the minimal effort involved, low overheads, and the strong incentives will motivate more ISPs and reflectors to participate in the Net-Police consortium.

Adversarial Impact on Convergence Time. The convergence time of Net-Police is unaffected by the presence of adversaries. This is because Net-Police never re-examines a subtree marked as benign (even if it was falsely implied due to a dishonest router) within the same Net-Police run. Instead, Net-Police converges at the link between the closest dishonest router (from the victim) and its honest parent router (identified as per Theorem 2), without exploring beyond them on the attack path. This strategy effectively prevents any adversary from delaying Net-Police convergence. In fact, when a router behaves dishonestly, Net-Police patrols a reduced search space, and therefore, converges even faster than usual. However, this may result in reduced benefits in terms of bandwidth savings but only to the ISP that owns the adversarial router.

Dynamic Adversary. During an ongoing Net-Police run, an attacker may move to another subtree that was marked benign in a previous cycle. However, as Net-Police does not re-examine the subtree within the same run, the adversary may escape the Net-Police patrol. However, synchronizing such a movement requires insights that can be categorized as privileged: **(1)** the start of Net-police patrol; **(2)** the knowledge of topology; **(3)** the PTs that were selected in previous cycles; **(4)** the presence/absence of other attackers; and, **(5)** the time period for which to remain quiet. Getting the information above is possible if and only if the attacker compromises an ISP on the path from a PT to the victim. Even in such cases, the probability of knowing such privileged information is very low. Specifically, the following aspects of Net-Police make identifying the PT difficult (Refer Section 5.3.1): **(1)** Use of a magic IP address in the source IP address field of the GRE packet; **(2)** Cryptographic infeasibility of mapping a label to the IP address of PT; and, **(3)** Use of encryption for control communication between the CA and victim as well as the CA and PT.

8. Conclusion

In this paper, we proposed Net-Police, a novel patrolling system for mitigating the impact of a volumetric DDoS attack at both the victim and the ISPs. Net-Police reliably identifies the sources of attack to facilitate quick filtering of the attack packets, even under adversarial conditions. We evaluated Net-Police in multiple real-world topologies, a physical testbed, as well as against a real-world attack scenario. Our experiments revealed that a strategic selection of routers using hybrid centrality metric, which uses a combination of betweenness and degree centrality, can assist in achieving quicker traceback and minimizing the overheads for the ISPs. To the best of our knowledge, this is the first work that benefits both the victim and ISPs at the same time, in terms of availability and ISP's bandwidth savings as compared to other state-of-the-art cloud-based and IP Traceback-based solutions.

Acknowledgement

This work is supported by Information Security Education and Awareness (ISEA) Project, Ministry of Electronics and Information Technology (MeitY), Govt. of India. The authors thank Balaji Venkat for his insightful comments and suggestions.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.comcom.2019.11.034.

References

[1] S. Mansfield-Devine, DDoS goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare, Netw. Secur. (11) (2016) 7–13.
[2] 602 Gbps!, this may have been the largest DDoS attack in history, 2018, URL https://thehackernews.com/2016/01/biggest-ddos-attack.html. (Accessed 2018-04-18).
[3] M. Anagnostopoulos, G. Kambourakis, P. Kopanos, G. Louloudakis, S. Gritzalis, DNS amplification attack revisited, Comput. Secur. 39 (2013) 475–485.
[4] C. Rossow, Amplification hell: Revisiting network protocols for DDoS abuse, in: NDSS, 2014.
[5] Imperva incapsula survey: What DDoS attacks really cost businesses, 2018, URL https://lp.incapsula.com/rs/804-TEY-921/images/eBook%20-%20What%20DDoS%20Attacks%20Really%20Cost%20Businesses%20%28new%29pdf. (Accessed 2018-09-01).
[6] The cost of downtime at the world's biggest online retailer, 2018, URL https://www.upguard.com/blog/the-cost-of-downtime-at-the-worlds-biggest-online-retailer. (Accessed 2018-09-13).
[7] V. Giotsas, P. Richter, G. Smaragdakis, A. Feldmann, C. Dietzel, A. Berger, Inferring BGP blackholing activity in the internet, in: Proceedings of the Internet Measurement Conference, ACM, 2017, pp. 1–14.
[8] DDoS Attacks on internet providers can impact downstream customers, 2018, URL https://www.corero.com/blog/840-ddos-attacks-on-internet-providers-can-impact-downstream-customers.html. (Accessed 2018-08-01).
[9] Z. Liu, H. Jin, Y. Hu, M. Bailey, Middlepolice: Toward enforcing destination-defined policies in the middle of the internet, in: ACM Conference on Computer and Communications Security, ACM, 2016, pp. 1268–1279.
[10] Protect against DDoS attack , 2018, URL https://www.cloudflare.com/ddos. (Accessed 2018-08-18).
[11] C. Schutijser, Comparing ddos mitigation techniques, 24th Twente Student Conference on IT, 2016.
[12] S. Savage, D. Wetherall, A.R. Karlin, T.E. Anderson, Network support for IP traceback, IEEE/ACM Trans. Netw. 9 (3) (2001) 226–237.
[13] R. Chen, J.-M. Park, Attack diagnosis: throttling distributed denial-of-service attacks close to the attack sources, in: 14th International Conference on Computer Communications and Networks, IEEE, 2005, pp. 275–280.
[14] L. Cheng, D.M. Divakaran, W.Y. Lim, V.L. Thing, Opportunistic piggyback marking for IP traceback, IEEE Trans. Inf. Forensics Secur. 11 (2) (2016) 273–288.
[15] R. Chen, J.-M. Park, R. Marchany, TRACK: A novel approach for defending against distributed denial-of-service attacks, Technical Report TR ECE-06-02, Dept. of Electrical and Computer Engineering, Virginia Tech, 2006.
[16] A. Belenky, N. Ansari, IP traceback with deterministic packet marking, IEEE Commun. Lett. 7 (4) (2003) 162–164.
[17] S. Yu, W. Zhou, S. Guo, M. Guo, A feasible IP traceback framework through dynamic deterministic packet marking, IEEE Trans. Comput. 65 (5) (2016) 1418–1427.
[18] S. Ramanathan, J. Mirkovic, M. Yu, Y. Zhang, SENSS against volumetric DDoS attacks, in: Proceedings of the 34th Annual Computer Security Applications Conference, ACM, 2018, pp. 266–277.
[19] V.A. Foroushani, A. Zincir-Heywood, TDFA: Traceback-based defense against DDoS flooding attacks, in: IEEE 28th International Conference on Advanced Information Networking and Applications, AINA, IEEE, 2014, pp. 597–604.
[20] AS Relationships, CAIDA, 2019, URL http://data.caida.org/datasets/as-relationships/. (Accessed 2019-03-15).
[21] The Internet Topology Zoo, 2018, URL http://www.topology-zoo.org/. (Accessed 2018-09-01).
[22] Mirai-like botnet - bad packets report, 2019, URL https://mirai.badpackets.net/. (Accessed 2019-02-17).
[23] Google cloud armor: Defending your services, 2019, URL https://cloud.google.com/armor/. (Accessed 2019-04-20).

[24] DDoS Protection, Akamai, 2019, URL https://www.akamai.com/us/en/resources/ddos-protection.jsp. (Accessed 2019-04-20).
[25] K. Singh, P. Singh, K. Kumar, A systematic review of IP traceback schemes for denial of service attacks, Comput. Secur. 56 (2016) 111–139.
[26] J. François, O. Festor, Anomaly traceback using software defined networking, in: 2014 IEEE International Workshop on Information Forensics and Security, WIFS, IEEE, 2014, pp. 203–208.
[27] H. Zhang, J. Reich, J. Rexford, Packet traceback for software-defined networks, Princeton University, 2015.
[28] P. Thai, J.C. de Oliveira, Decoupling policy from routing with software defined interdomain management: Interdomain routing for SDN-based networks, in: 22nd International Conference on Computer Communication and Networks, ICCCN, IEEE, 2013, pp. 1–6.
[29] R. Varloot, BGP-Level Topology of the Internet: Inference, Connectivity and Resiliency, Tech. rep., 2013.
[30] R. Bennesby, E. Mota, A survey on approaches to reduce BGP interdomain routing convergence delay on the internet, IEEE Commun. Surv. Tutor. (2017).
[31] A. Wang, W. Chang, A. Mohaisen, S. Chen, POSTER: How distributed are today's DDoS attacks? in: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 1511–1513.
[32] OVH hosting hit by 1Tbps DDoS attack, the largest one ever seen, 2018, URL http://securityaffairs.co/wordpress/51640/cyber-crime/tbps-ddos-attack.html. (Accessed 2018-04-18).
[33] Interpol, 2019, URL https://www.interpol.int/en. (Accessed 2019-08-25).
[34] World trade organization, 2019, URL https://www.wto.org/. (Accessed 2019-08-25).
[35] World health organization, 2019, URL https://www.who.int/. (Accessed 2019-08-25).
[36] S. Newman, Service providers: the gatekeepers of internet security, Netw. Secur. 2017 (5) (2017) 5–7.
[37] R. Motamedi, R. Rejaie, W. Willinger, A survey of techniques for internet topology discovery, IEEE Commun. Surv. Tutor. 17 (2) (2014) 1044–1065.
[38] D. Samociuk, Secure communication between openflow switches and controllers, in: The Seventh International Conference on Advances in Future Internet, AFIN, vol. 39, 2015.
[39] RTT distribution, 2018, URL http://www.caida.org/projects/ark/statistics/eug-us.html. (Accessed 2018-06-30).
[40] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker, Controlling high bandwidth aggregates in the network, ACM SIGCOMM Comput. Commun. Rev. 32 (3) (2002) 62–73.
[41] S.J. Templeton, K.E. Levitt, Detecting spoofed packets, in: Proceedings DARPA Information Survivability Conference and Exposition, Vol. 1, IEEE, 2003, pp. 164–175.
[42] Default TTL values in TCP/IP, 2018, URL http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm#detailed. (Accessed 2018-08-21).
[43] U. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 (2) (2001) 163–177.
[44] S.P. P. Phaal, N. McKee, InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, IETF, 2001, URL https://tools.ietf.org/html/rfc3176. (Accessed 2018-08-01).
[45] E. B. Claise, Cisco Systems NetFlow Services Export Version 9, IETF, 2001, URL http://netflow.caligare.com/rfc_3954.txt. (Accessed 2018-08-01).
[46] P. Sattari, M. Gjoka, A. Markopoulou, A network coding approach to IP traceback, in: IEEE International Symposium on Network Coding, NetCod, IEEE, 2010, pp. 1–6.
[47] V. Paruchuri, A. Durresi, S. Chellappan, TTL based packet marking for IP traceback, in: IEEE Global Communications Conference, GLOBECOM, 2008, pp. 2552–2556.
[48] L. Lu, M.C. Chan, E.-C. Chang, A general model of probabilistic packet marking for IP traceback, in: Proceedings of ACM symposium on Information, Computer and Communications Security, ASIACCS, 2008, p. 179.
[49] C. Gong, K. Sarac, Toward a practical packet marking approach for IP traceback, Ger. J. Agric. Econ. 62 (2013) 271–281.
[50] V.A. Foroushani, A. Zincir-Heywood, Deterministic and authenticated flow marking for IP traceback, in: 27th International Conference on Advanced Information Networking and Applications, AINA, IEEE, 2013, pp. 397–404.
[51] P. Wu, Y. Cui, J. Wu, J. Liu, C. Metz, Transition from IPv4 to IPv6: A state-of-the-art survey, IEEE Commun. Surv. Tutor. 15 (3) (2013) 1407–1424.
[52] D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, Generic Routing Encapsulation (GRE), IETF, 2000, URL https://tools.ietf.org/html/rfc2784. (Accessed 2018-09-01).
[53] G. Dommety, Key and Sequence Number Extensions to GRE, IETF, 2000, URL https://tools.ietf.org/html/rfc2890. (Accessed 2018-09-01).
[54] S. Raman, B. Venkat, G. Raina, Mitigating some security attacks in MPLS-VPN model "C", Int. J. Adv. Netw. Serv. 5 (3 & 4) (2012).
[55] N.F. Pub, Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication, 2001, URL http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. (Accessed:2018-08-01).
[56] X. Liu, X. Yang, Y. Xia, Netfence: Preventing internet denial of service from inside out, in: ACM SIGCOMM Computer Communication Review, vol. 40, ACM, 2010, pp. 255–266.
[57] How secure is AES against brute force attacks? 2018, URL https://www.eetimes.com/document.asp?doc_id=1279619. (Accessed 2018-08-01).
[58] X. Zhang, S.F. Wu, Z. Fu, T.-L. Wu, Malicious packet dropping: how it might impact the TCP performance and how we can detect it, in: International Conference on Network Protocols, IEEE, 2000, pp. 263–272.
[59] Packet size distribution comparison between internet links in 1998 and 2008, 2018, URL https://www.caida.org/research/traffic-analysis/pkt_size_distribution/graphs.xml. (Accessed 2018-08-21).
[60] L. Su, D.M. Divakaran, V. Thing, Privacy preserving IP traceback, in: IEEE 4th International Conference on Identity, Security, and Behavior Analysis, ISBA, IEEE, 2018, pp. 1–8.
[61] R. Castro, M. Coates, G. Liang, R. Nowak, B. Yu, Network tomography: Recent developments, Statist. Sci. (2004) 499–517.
[62] Attack signatures, 2018, URL https://www.symantec.com/security_response/attacksignatures/, (Accessed 2018-09-1).
[63] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, et al., Understanding the mirai botnet, in: 26th USENIX Security Symposium, 2017, pp. 1093–1110.
[64] A quick stats on the 608,083 Mirai IPs that hit our honeypots in the past 2.5 months, 2019, URL https://blog.netlab.360.com/a-quick-stats-on-the-608-083-mirai-ips-that-hit-our-honeypots-in-the-past-2-5-months/. (Accessed 2019-02-17).
[65] Intel Galileo, URL https://www.arduino.cc/en/ArduinoCertified/IntelGalileo. (Accessed 2018-09-01).
[66] Quagga Routing Suite, URL http://www.nongnu.org/quagga/. (Accessed 2018-08-01).
[67] Scaling in the Linux Networking Stack, URL https://www.kernel.org/doc/Documentation/networking/scaling.txt. (Accessed 2018-09-1).
[68] Cisco 2016 Annual Security Report, URL http://www.cisco.com/c/dam/assets/offers/pdfs/cisco-asr-2016.pdf. (Accessed 2018-08-18).