Linking Amplification DDoS Attacks to Booter Services

Johannes Krupp[1(✉)], Mohammad Karami[2], Christian Rossow[1], Damon McCoy[3], and Michael Backes[1,4]

[1] CISPA, Saarland University, Saarbrücken, Germany
`johannes.krupp@cispa.saarland`
[2] Google, Inc., Mountain View, CA USA
`mkarami@google.com`
[3] New York University, New York, USA
[4] MPI-SWS, Saarbrücken, Germany

Abstract. We present techniques for attributing amplification DDoS attacks to the booter services that launched the attack. Our k-Nearest Neighbor (k-NN) classification algorithm is based on features that are characteristic for a DDoS service, such as the set of reflectors used by that service. This allows us to attribute DDoS attacks based on observations from honeypot amplifiers, augmented with training data from ground truth attack-to-services mappings we generated by subscribing to DDoS services and attacking ourselves in a controlled environment. Our evaluation shows that we can attribute DNS and NTP attacks observed by the honeypots with a precision of over 99% while still achieving recall of over 69% in the most challenging real-time attribution scenario. Furthermore, we develop a similarly precise technique that allows a victim to attribute an attack based on a slightly different set of features that can be extracted from a victim's network traces. Executing our k-NN classifier over all attacks observed by the honeypots shows that 25.53% (49,297) of the DNS attacks can be attributed to 7 booter services and 13.34% (38,520) of the NTP attacks can be attributed to 15 booter services. This demonstrates the potential benefits of DDoS attribution to identify harmful DDoS services and victims of these services.

1 Introduction

Distributed Denial-of-Service (DDoS) attacks have become commoditized by DDoS-for-hire services, commonly called *booters* or *stressers* [7,19]. A large number of booter services advertise their services openly as an economical platform for customers to launch DDoS attacks. At the same time DDoS attacks are increasing in number and in magnitude. This proliferation of DDoS attacks has caused many network and website operators to rank this type of attack as one of the largest threats facing them [13]. This barrage of DDoS attacks has increased the demand for Content Delivery Networks (CDNs) and Software Defined Networking defenses that can absorb and filter these attacks [5]. In turn, this has prompted attackers to react by devising increasingly efficient methods of bypassing or overwhelming defenses. The result is an escalating technological arms-race between DDoS attackers and defenders that at times has congested segments of the core Internet infrastructure as collateral damage [17].

Despite the proliferation of DDoS services and attacks, little progress has been made on attributing the services that are launching these attacks on behalf of their customers. Most ideas for attribution focus on IP traceback mechanisms [16, 21–23, 30] to trace the source of spoofed IP packets, which require ISPs to assist and so far have not been widely deployed. This has resulted in most of these attacks being unattributed unless the attackers unveil themselves. While it is important to create strong technological DDoS defenses, we argue that there is also benefit in investigating other methods that enable attribution of DDoS attacks to the services responsible for launching these attacks. For instance, some of these booter services—seven out of 23 services that we studied—claim they are benign services by advertising as "stress-testing" services intended to be used only by authorized administrators. For example, one of these services included this statement on their website, *"We provide a professional and legal ip stresser service which is based on a massive 20 dedicated server backend, ensuring that your server is tested to its limits."* Attribution can remove this veil of legitimacy and assist efforts to undermine these services by allowing victims and law enforcement to attribute which booter services were responsible for an attack. Attribution also enables measuring the scale of these services and prioritizing undermining the larger services that are causing more harm. In order to assist ongoing investigations, we are continually sharing information from our study on DDoS attacks and booter services with the European Police Office (Europol), the United States Federal Bureau of Investigation (FBI) and large ISPs or backbone providers.

In this work, we show that it is possible to build supervised learning techniques that allow honeypot amplifier operators and victims to accurately attribute attacks to the services that launched them. To begin, we identify three key features that honeypot operators can record to construct a supervised k-NN classifier that can attribute attacks. In order to validate our method, we subscribed to 23 booter services and generated a ground truth data set of attacks to booter service mappings[5]. Validation of our classifier using the ground truth self-attack data set shows that it is highly precise at attributing DNS and NTP attacks with a precision of over 99% at 69.35% recall in the worst case of real-time attribution. When retrospectively attributing attacks, the recall even increases to 86.25%. Executing our classifier over the set of all attacks observed by the honeypots shows that 25.53% (49,297) of the DNS attacks can be attributed to 7 booter services and 13.34% (38,520) of the NTP attacks can be attributed to 15 booter services.

Finally, we show that a k-NN classifier can also be used *by victims* to attribute DDoS attacks to the service that launched the attack. Our findings demonstrate

[5] Our ethical framework for these measurements is based on previous studies that have used this methodology [7, 20].

that many of the attacks we observed can be attributed to a small set of booter services that are operating relatively openly. Our ability to attribute large numbers of attacks to a small set of booter services and sharing of this information with Europol and the FBI to assist in active investigations demonstrates the usefulness of our attribution methods.

In summary, we frame our contributions as follows:

– We present a k-NN-based classifier that attributes amplification DDoS attacks observed by honeypots with a precision of over 99% while still achieving recall of over 69% in the most challenging real-time attribution scenario.
– We present a similarly precise technique that allows a DDoS victim to attribute attacks based on features extracted from a victim's network traces.
– We attribute 25.53% (49,297) of the DNS attacks to 7 booter services and 13.34% (38,520) of the NTP attacks to 15 booter services.

2 Background

2.1 Threat Model

Amplification DDoS constitutes a powerful attack in which an adversary aims to exhaust the bandwidth of a victim's host or network by inducing a large volume of traffic. Towards this, the attacker abuses multiple servers as so called *amplifiers*. These servers offer UDP-based protocols prone to amplification, i.e., the server's response is significantly larger than the corresponding request sent to the server. At least 14 protocols suffer from this flaw [18], such as NTP and DNS, leading to a multitude of servers that can be exploited as amplifiers. Given the connection-less nature of UDP, an attacker can redirect the servers' responses to the victim by simply spoofing the source IP address in requests. Due to amplification ratios of a factor of 5 to 4500 [18], an attacker that sends requests at a rate of some Mbit/s can still cause attack traffic at Gbit/s-scale.

Furthermore, we are concerned with a special type of attacker: *booter services*. These offer platforms for DDoS-as-a-service, often under the disguise of "stress-testing", where customers can request various types of attacks for a small fee. The booter will then launch these attacks using its infrastructure. Our threat model thus contemplates four parties: *Customers*, who commission attacks; *booters*, who conduct the actual attacks; *amplifiers*, who are exploited to amplify traffic; and *victims*, who are the targets of such attacks.

The aim of this paper is to attribute attacks to booters, when observed from either the victim's or an amplifier's perspective. This is non-trivial, as from the victim's perspective the attack seems to stem from the amplifiers. Similarly, from an amplifier's perspective, the requests seem to be legitimate requests by the victim (due to the use of spoofed source IP addresses by the booter). While ultimately one would like to identify the customer, only the booter, amplifiers, and the victim are directly participating in an attack. Nonetheless, since the booter has a business relation to the customer, pinpointing the booter behind an attack constitutes an important step towards this goal.

2.2 Ethical Considerations

As part of our study we subscribed to 23 booters and conducted a controlled set of self DDoS attacks. Furthermore, we also leveraged honeypots for amplification attacks. We settled on this methodology for collecting a ground truth data set of mappings between observed attacks and the services that launched these attacks after finding that no data set available to us could be used to validate our DDoS attribution techniques. Before we began performing these self DDoS attacks we carefully attempted to minimize the harms and maximize the benefits associated with our methodology based on observations from previous studies that launch self-attacks in order to measure booter's attacks [7, 20].

We received an exemption from our Institutional Review Board (IRB), since our study did not include any personally identifiable information. In addition, we consulted with our institution's general counsel, who advised us not to engage with any DDoS service that advertised using botnets and to cease active engagement with any booter service that we realized was using botnets.

An analysis of TTL values observed by the honeypots indicated that it is unlikely any of the booter services we subscribed to used botnets. Based on the guidance of our institution's general counsel, our victim server was connected by a dedicated 1 Gbit/s network connection that was not shared with any other servers. We also obtained consent from our ISP and their upstream peering points before conducting any DDoS attack experiments. We also minimized the attack durations, notified our ISP before launching any attack and had a protocol in place to end an attack early if it caused a disruption at our ISP.

We purchased subscriptions from 23 booter services. When doing so, we selected the cheapest option, which ranged from \$6-\$20 and averaged \$12 per month, to minimize the amount of money given to these services. In total, we spent less than \$400 and no individual booter service received more than \$40 in payments as part of the measurements in this paper[6]. All payments were made using PayPal and we assumed that proper controls were put in place at PayPal to mitigate the risk of money flowing to extremist groups. As part of our design methodology, we minimized the amount of money paid and targeted a small set of booters to obtain a valuable ground truth data set.

Our method created some harm to amplifiers and their upstream peering points by consuming bandwidth resources. The largest amount of bandwidth consumed was 984.5 kbit/s for NTP amplifiers and the least was 16.7 kbit/s for DNS amplifiers, similar to those reported in a previous study [7].

Over the course of our experiments we did not receive any complaints from the operators of these amplifiers. We limited our attacks to 30 seconds. Based on analysis from a previous study that used a similar methodology [7], these short duration attacks enable us to observe about 80% of the amplifiers used by a given booter service and reduce the harm we cause to misconfigured amplifiers.

[6] To put this into perspective: Previous studies of these booters have shown that they have thousands of paid subscribers and generate revenues of over \$10,000 per month [7, 19].

Similarly, the use of DDoS honeypots might also incur harm on the Internet. We used AmpPot, a honeypot proposed by Kraemer et al. [8]. To avoid contributing to DDoS attacks, AmpPot limits the rate of requests and deploys automatic IP blacklisting: The honeypots will stop responding for one hour to any IP address sending more than 10 requests per minute. This limits the maximum amount of data sent to a DDoS victim to a few kilobytes. For a more detailed ethical discussion on AmpPot we refer the reader to [8].

3 Amplification Attack Data Set

To investigate if and how amplification attacks can be attributed to their originating booter service, we established two data sets that help us to gain insights into the overall amplification attacks, but also to find concrete attack instances caused by individual booters. In Section 3.1, we describe how we leverage amplification honeypots to gain insights into global amplification attacks. In Section 3.2, we discuss how we use booters and launch controlled attacks against ourselves to learn about attack techniques of certain attackers.

3.1 Honeypot Attacks

Although the general threat of amplification attacks has been known for years, actual attack insights are only documented in anecdotal evidence, such as attacks against Spamhaus or OVH at hundreds of Gbit/s attack volume. To collect insights into the set of global amplification attacks, we leverage data collected by AmpPot [8], a honeypot proposed by Krämer et al. AmpPot emulates seven UDP-based protocols that have known amplification vectors and will thus eventually be abused as part of real-world DDoS amplification attacks (QOTD, CharGen, DNS, NTP, RIPv1, MSSQL, and SSDP). Krämer et al. observed that attackers will eventually find such honeypots via Internet scans, and start abusing them as potential reflectors shortly thereafter. AmpPot thus serves as an eye on global amplification attacks, and due to the nature of the attack traffic, can also observe who is being attacked and when.

In December 2014, eleven globally-distributed honeypots with single static IP addresses were deployed, and have been operated continuously since then. In November 2015, a twelfth honeypot was added, listening on 48 static IP addresses. This honeypot employs a special feature named *Selective Response*, where each source scanning for amplifiers will find a unique set of 24 IP addresses[7] [10].

We set our analysis period to two months from December 9, 2015 to February 10, 2016. In this period, the honeypots observed 570,738 amplification attacks (8,918 attacks per day on average). However, given that RIPv1, MSSQL, and QOTD combined account for less than 5% of these, we decided to exclude those protocols from our analyses.

[7] The idea behind this is to imprint a unique fingerprint on each scanner. Letting each scanner find 24 IP addresses maximizes the total number of fingerprints.

Table 1: Covered booter services

	AUR	BAN	BO1	BO2	BO3	CRI	DOW	EXI	EXO	KST	INB	NET	RAW	SER	STA	ST1	ST2	ST3	ST4	SYN	THU	VDO	WEB
CharGen	✓	✓	✓	✓	✓		✓		✓	✓		✓	✓	✓					✓	✓	✓		✓
DNS	✓	✓		✓	✓		✓	✓			✓		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
NTP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
SSDP	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓			✓	✓	✓		✓		✓	✓	

3.2 Self-Attacks

The honeypots give us valuable insights into global attacks, but do not give us indications where the attacks were coming from. Previous studies have identified so-called booter services ("booters") as being responsible for a large number of amplification attacks [6, 7, 20]. In an attempt to learn attack characteristics of these booters, we signed up at these services and then launched short-lived amplification attacks against a target in our control.

To start launching self-attacks and correlating them with the traffic seen at the honeypots, our first task was to identify booter services to cover in the study. Absent a centralized location for finding booters, we located services via search engines and advertisements on underground forums. We selected a total of 23 services offering amplification attacks based on NTP, DNS, CharGen and SSDP. When selecting these booters, we tried to include services that we speculated to be more stable and have more subscribers based on reviewing user feedback on underground forums. To minimize the amount of money we paid to these abusive services, we kept the number of covered booters relatively small.

Table 1 provides an overview of the booter services[8] that we cover and the amplification attack types they offer. NTP was the most popular attack protocol, followed by DNS. 16 of the 23 services clearly advertise malicious DDoS attacks. In contrast, seven services hide their malicious intention behind "stressing" services, a seemingly benign way to verify the resilience of a network against DDoS attacks. However, not a single service performs any kind of attack target validation. That is, service subscribers can specify any IP address (or domain) that should be attacked, regardless of whether the target is under the control of the client. This shows the clear malicious intention behind all 23 booter services.

Booter services maintain front-end sites that allow their customers to purchase subscriptions and launch attacks using simple web forms. We created custom crawlers to automate the task of visiting the websites of covered booters and launching attacks directed at our own target. Using this automation, daily attacks were launched for each covered booter and attack type. A total of 13 booter services were covered within the first week of starting the self-attacks on December 9, 2015 and by January 14, 2016 all 23 booters were covered.

[8] To avoid unintentionally advertising booter services covered in this study, we replace the name of booter services by the first three letters of their domain name. The last letter is replaced by a number in the case of name collisions.

Table 2: Overview over self-attacks

Protocol	**booters**	**attacks launched**	**observed at victim**	**observed at honeypots**	**observed >100 pkts.**
CharGen	16	608	417	35	33
DNS	19	676	452	173	100
NTP	22	823	577	421	373
SSDP	16	560	351	1	0
Total	23	2667	1797	630	506

Labeling Self-Attacks As we instructed all booters to attack the same target, we had to find a mechanism to separate between multiple consecutive self-attacks to assign (booter) labels to the attack traffic. To this end, we initially relied on the time that attacks were initiated. To account for clock skew, we left 10 minutes between consecutive attacks and used a grace period of ±3 minutes for matching. On January 14, we started to use a distinct victim IP per booter service as an improved matching criterion. Based on the same criterion, we then also mapped the self-attacks to attacks observed at the honeypots.

Table 2 gives an overview over the self-attacks. We launched a total of 2667 CharGen, DNS, NTP and SSDP attacks using 23 booter services. Interestingly, only around 2/3 of the attacks we initiated were observed at the victim. This can be explained by our observation of maintenance issues that some booter websites have. Sometimes booter websites provide the user interface for selecting a particular attack type that is temporarily non-functional. To users it appears that the attack has been successfully launched, but no actual attack traffic is generated as a result of initiating such attacks.

The DDoS honeypots observed many NTP attacks (73.0%) and DNS attacks (38.3%), but only a small fraction of the CharGen attacks (8.4%) and only a single SSDP attack. Furthermore, while the honeypots observed some traffic belonging to 630 attacks, in only 506 cases did we record more than 100 requests. We inspected the reasons why the honeypots missed large portions of SSDP and CharGen attacks. To this end, we investigated the attack traffic towards our victim to learn the preferences of attacks in choosing reflectors. In both cases, we found that the vast majority of the reflectors that were abused by multiple booters send responses that are significantly larger than the ones configured in AmpPot. This indicates that the honeypots' SSDP and CharGen responses were too small to be attractive for attackers, and adversaries preferred other reflectors with better amplification. We leave further investigations on reflector selection strategies open for future work and focus on DNS and NTP in the following.

Multi-Branding Booters During the sign-up phase, we noticed that some booters were visually similar. Investigations have revealed that one miscreant follows a multi-branding strategy, i.e., sells the same service via different booter names that shared similar web front-ends. It became apparent that attacks from `RAW` and `WEB` shared characteristics, and also their sign-up page of the web interface was equivalent in appearance and HTML code. We further analyzed those two booters by launching application layer (layer 7) attacks against our victim server. Layer 7 attacks usually abuse public HTTP proxy servers to hide the identity of back-end servers involved. However, some proxies reveal the identity of the requesting clients in the `X-Forwarded-For` field of the HTTP header. Based on this observation, we were able to verify that these two booters used shared back-end infrastructure. We thus conclude that `RAW` and `WEB` are likely to be operated by the same individuals and will regard them as equivalent.

4 Characteristic Attack Features

We will now introduce characteristic attack patterns that we can use to train our classifier for attribution purposes. We first describe various characteristics that we have observed to repeat across subsets of attacks at the honeypots. We then describe how we leverage these observations as features to summarize attacks.

4.1 Attack Observations

While analyzing the attacks captured by the honeypots, we observed the following three properties that repeated across subsets of the attacks.

Honeypot Sets: Although eleven honeypots were active since the end of 2014, few attacks (1.63%) abused all of them simultaneously. In fact, more than 60% of all DNS- and NTP-based attacks abused five honeypots or less. This indicates that attackers either perform only partial scans of the Internet, or choose a subset of the discovered amplifiers in subsequent attacks.

Interestingly, we observed that honeypot sets seem to be *reused* across multiple attacks, i.e., even in attacks against different victims or on different days. To further investigate this observation, we analyzed amplifiers seen in self-attacks from a few example booter services over time, shown in Figure 1. The entries on the heat maps show the ratio of abused amplifiers that were shared per booter and attack protocol on two consecutive days each. With the exception of DNS, there is a high level of overlap for attacks based on NTP, CharGen, and SSDP, suggesting that booters reuse their set of amplifiers for a protocol for some time. The low overlap for attacks based on DNS is likely caused by frequent rescans to account for the relatively high IP churn rate of DNS amplifiers [11].

In addition, we verified that two simultaneous attacks towards the same victim on different protocols showed little overlap in the sets of honeypots abused. This could indicate that the set of amplifiers might be specific to the protocol, which intuitively can be explained by the small overlap of systems that suffer from amplification vulnerabilities for multiple protocols.

Victim Ports Entropy: While one UDP port determines the amplification protocol (e.g., DNS, NTP, etc.), the other determines the *victim port* on which the victim will receive the reflected responses. Since an attacker has virtually no restrictions on setting the victim port, we expected to observe the two obvious choices: Choosing one victim port per attack, or choosing an individual victim port for every request. Surprisingly, in addition to that, we also observed attacks

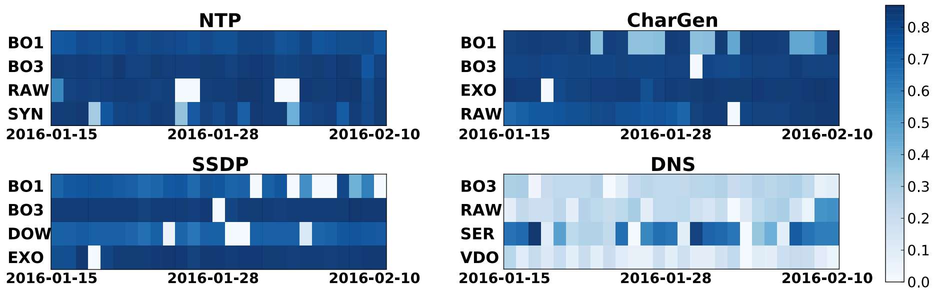

Fig. 1: Overlap of amplifier sets between consecutive dates.

where requests shared a small number of victim ports. One explanation could be that attackers use multiple threads for attacking, and that they choose a different victim port *per thread*. In addition, we verified that a significant number of booter services actually ask their clients to choose the victim port, giving a reason why the number of source ports is frequently restricted to one.

Time-to-Live Values: The Time-to-Live (TTL) value in the IP packet indicates how many hops a packet has traversed from the attack source to the honeypot. As already observed by Krämer et al. [8], for one particular attack, a honeypot will usually only see one (or very few) TTL value(s). We can thus conclude that most attacks likely stem from a single source, which motivates further investigations in finding this particular source sending spoofed traffic. Additionally, the vast majority of requests have a TTL > 230. This suggests that attackers use a fixed initial TTL of 255 in their generated packets, as otherwise we would see a wider distribution.

4.2 Distance Function

In order to leverage these observations in a classifier, we next introduce a distance function based on the above features. Given two attack instances A and B, such a function is used to determine how dissimilar the two instances are. For an attack A, we will denote the set of honeypots used by $\mathbf{HP}_A$, the set of victim ports observed by $\mathbf{VPort}_A$, and the set of TTLs received at honeypot hp by $\mathbf{TTL}_{hp,A}$.
To compare honeypot sets, we leverage the well-known Jaccard distance:

$$d_{hp}(A,B) = 1 - \frac{|\mathbf{HP}_A \cap \mathbf{HP}_B|}{|\mathbf{HP}_A \cup \mathbf{IP}_B|}$$

To compare the set of victim ports, we take the normalized difference:

$$d_{vp}(A,B) = \frac{\big||\mathbf{VPort}_A| - |\mathbf{VPort}_B|\big|}{\max\left(|\mathbf{VPort}_A|, |\mathbf{VPort}_B|\right)}$$

Finally, to compare TTLs, we compute the overlap of their histograms[9]

$$d_{\text{hist}}(S,T) = 1 - \frac{\sum\limits_x \min(S(x),T(x))}{\sum\limits_x \max(S(x),T(x))}$$

and then average this overlap over all honeypots involved in *both* attacks:

$$d_{ttl}(A,B) = \frac{\sum\limits_{hp \in \mathbf{HP}_A \cap \mathbf{HP}_B} d_{\text{hist}}\left(\mathbf{TTL}_{hp,A}, \mathbf{TTL}_{hp,B}\right)}{|\mathbf{HP}_A \cap \mathbf{HP}_B|}$$

From these three sub-functions we compute a weighted average as the overall distance function. We set the weights to $w_{hp} = 5$, $w_{vp} = 1$, and $w_{ttl} = |\mathbf{HP}_A \cap \mathbf{HP}_B|/2$. Note that our methodology is independent from the weights and the analyst can choose any weights according to her needs. We assigned a smaller weight to the victim port feature, as it relies on inputs with little entropy given just three cases: a single victim port, a few victim ports, or many victim ports. For the TTL feature, we assign a higher weight if the two attacks have more honeypots in common, as we assume that coinciding TTLs for *multiple* honeypots have a much higher significance than those for only a single honeypot.

5 Honeypot Attack Attribution

We now leverage the aforementioned features to identify which booter has caused which attacks observed at a honeypot. The core idea is to use supervised machine learning techniques to attribute an attack observed at a honeypot to a particular booter service. We will first use our ground truth data set to show the performance and resilience of our classifier in various situations. Afterwards, we will apply the classifier to the entire data set of attacks collected by the honeypots.

5.1 Description

Finding the true origin of an amplification attack is a non-trivial problem, because—from the reflector's perspective—all packets carry spoofed headers. Using our attack distance metric, we showed that attacks from the same booter service exhibit similar characteristics and this observation turns the problem of finding the origin of an attack into a classification problem. The collected self-attack data set can be used for training and validating a classifier. Since the number of attacks observed strongly varies between booters, we decided to use the *k-Nearest Neighbor* (k-NN) algorithm due to its resilience to such imbalances. In k-NN, to determine the label of an instance, the set of its k nearest

[9] To account for fluctuation in TTLs due to route changes, we apply smoothing to the histograms using a binomial kernel of width 6, which corresponds to a standard deviation of $\sigma \approx 1.22$.

neighbors is computed. Next, every neighbor casts a vote for its own label, and finally the instance is given the label of the majority of its neighbors.

Additional care has to be taken, as our training data set is not exhaustive and may miss data for some booters. That is, not all attacks can be attributed to a booter that we know. Therefore, we use a cutoff threshold t to introduce a label for an `unknown` classification result. When classifying an item i, we only consider the k nearest neighbors that can be found in the neighborhood of radius t centred around item i. If no item from the training data set lies within this neighborhood, the item i is assigned the label `unknown`. To find a well-suited and conservative threshold, we analyzed our ground truth data set using our distance function and hierarchical clustering. From those clusters, we then computed the average distance between attacks within a cluster and took the 95th percentile over all. This results in $t = 0.338$ for DNS and $t = 0.236$ for NTP.

Furthermore, as shown in Section 4.1, booters rescan to find new lists of amplifiers on a regular basis. To reflect this during classification, we only consider elements from the training data set no more than 7 days apart, which approximately corresponds to the maximum rescan frequency we observed for booters.

When using k-NN, the choice of k is highly critical for the performance of the classifier. One common approach is to learn the value of k from the training data set using *n-fold cross-validation* (CV). In n-fold CV, the training data set is partitioned into n equally sized sets. Then, the classifier is trained on $n - 1$ of these sets, and the final set is used for validation. This process is repeated n times, until every set has been used as the validation set once. For finding k we thus perform 10-fold CV for all $k \in \{1, 3, 5\}$ as part of the training phase of the classifier. We restrict k to odd values to avoid ties in the voting phase. We only consider $k \leq 5$, because about $^2/_3$ of the clusters contain less than five attacks.

To assess the performance of our classifier, we first define the *false positive rate* (FPR), *precision* and *recall* metrics, as well as *macro-averaging*. Intuitively, the FPR for a label l_i (in our case, a particular booter) is the fraction of elements that were incorrectly assigned the label l_i while their true label was *not* l_i. In a similar vein, precision is the ratio with which the classifier was correct when assigning label l_i, while recall is the ratio with which the classifier is able to re-identify elements with true label l_i. Let tp_i be the number of items *correctly* classified to have label l_i (*true positives*), let tn_i be the number of items *correctly* classified to *not* have label l_i (*true negatives*), let fp_i be the number of items *incorrectly* classified to have label l_i (*false positives*), and let fn_i be the number of items *incorrectly* classified to *not* have label l_i (*false negatives*). Then the FPR is defined as $\mathsf{fpr}_i = \mathsf{fp}_i/(\mathsf{fp}_i + \mathsf{tn}_i)$, precision as $\mathsf{p}_i = \mathsf{tp}_i/(\mathsf{tp}_i + \mathsf{fp}_i)$, and recall as $\mathsf{r}_i = \mathsf{tp}_i/(\mathsf{tp}_i + \mathsf{fn}_i)$. To compute overall performance measures from these per-class metrics, we employ macro-averaging, i.e., first computing fpr, p, and r *per class* and averaging the respective results afterwards, as this will avoid bias due to imbalance in our ground truth data. Thus booters for which we were able to collect more datapoints do *not* influence the results more strongly. However, since

Table 3: Honeypot-Driven Experimental Results

(a) DNS

		BAN	EXI	RAW	SER	ST1	ST2	VDO
	samples (#)	10	1	49	11	10	18	1
E1	correct (%)	90	0	82	82	100	78	0
E1	unknown (%)	10	100	18	18	0	22	100
E1	wrong (%)							
E2	unknown (%)	100	100	100	100	100	100	100
E2	wrong (%)							
E3	correct (%)	70	0	67	73	70	67	0
E3	unknown (%)	30	100	33	27	30	33	100
E3	wrong (%)							

(b) NTP

		AUR	BAN	BO1	BO2	BO3	CRI	DOW	EXI	NET	RAW	SER	ST1	ST3	ST4	SYN	THU	VDO
	samples	28	15	40	1	12	27	21	4	1	78	16	19	18	24	5	1	63
E1	correct	100	87	78	0	100	96	90	50	0	99	100	100	94	100	80	0	100
E1	unknown	0	13	23	100	0	4	10	50	0	1	0	0	6	0	20	100	0
E1	wrong									CRI: 100								
E2	unknown	100	100	100	100	100	74	100	100	0	100	100	100	100	100	100	100	100
E2	wrong						NET: 26			CRI: 100								
E3	correct	89	67	57	0	92	81	76	25	0	88	88	84	78	92	60	0	97
E3	unknown	11	33	43	100	8	19	24	75	0	12	13	16	22	8	40	100	3
E3	wrong									CRI: 100								

we strongly prefer mislabeling an attack as `unknown` over incorrectly attributing it to a wrong booter, we only weigh the `unknown` label with $\frac{1}{8}$.

5.2 Validation

To validate our classifier, we defined three experiments on our labeled self-attack data set: First, we conducted 10-fold CV to assess how well our classifier can correctly attribute attacks (E1). Second, to estimate how well our classifier deals with attacks from booters *not* contained in the training data set, we used leave-one-out CV on the booter level (E2). This means that the attacks from all but one booter constitute the training set, and all attacks from the omitted booter are used for validation, checking if these attacks are correctly labeled as `unknown`. Third, we were also interested in the performance of classifying attacks in real-time (E3), i.e., training only on labeled observations *prior* to the attack.

Table 3 shows the results for both DNS and NTP. For each experiment we give the percentage of attacks correctly attributed to its booter, the percentage of attacker the classifier labeled as `unknown`, as well as the percentage of attacks that were misclassified, along with their putative label. Additionally, the first row states the number of attacks contained in our data set[10]. Note that in the second experiment (E2) every column regards a classifier trained on the entire data set *except* the corresponding booter; hence the classifier is correct when assigning the `unknown` label in this case.

[10] This effectively provides the entire confusion matrix for each experiment.

In the 10-fold CV (E1) our DNS classifier correctly attributed 78% or more of the attacks for each booter. Exceptions are the cases of `EXI` and `VDO`, for which our data set only contains a single attack, which naturally cannot be attributed correctly due to lack of training data. All the remaining attacks were labeled as `unknown`. In fact, the DNS classifier never attributed an attack to a wrong booter in all three experiments. This is especially remarkable in the leave-one-out scenario (E2), when the classifier was not trained on data for one of the booters. That is, even in this case our classifier did not lead to false accusations, showing the resilience of the classifier against attacks stemming from booters *not* contained in the training set. Of course, this resilience comes at the cost of higher false negative rates in the other experiments (E1 & E3), as we prefer the classifier to label an attack as `unknown` over attributing it to the wrong booter. This could possibly be alleviated by obtaining more training data per booter. The last experiment (E3) simulates the performance of the classifier in a real-time scenario, i.e., when classifying an attack only based on training data that was obtained *prior* to the attack. In contrast to this, the first experiment (E1) measured the performance when classifying attacks after the fact. Since booters regularly rescan for amplifiers and update their set of amplifiers accordingly, our classifier will achieve a performance worse than in the first experiment (E1). However, even in the real-time attribution setting, we could still attribute at least 67% of all attacks without any incorrect attributions. The loss compared to E1 can be explained by the fact that the first attack of a booter can never be correctly classified due to lack of prior training data.

In the case of NTP, we achieved an overall attribution rate of 78% or more in the 10-fold CV (E1) for most booters, with the exception of those which occur only once in the data set. Remarkably, the cases of `EXI` and `SYN` show that the classifier also performs reasonably well even for small amounts of training data. The NTP classifier generates misclassifications. However, this only stems from a few attacks by `NET` and `CRI`, which exhibit precisely the same characteristics. While we suspect that `NET` and `CRI` share the same infrastructure, we were not able to verify this assumption by leveraging layer 7 attacks (as done previously for `RAW` and `WEB`). The same two attacks are also the cause for the only misclassifications in the leave-one-out scenario (E2), as about a quarter of attacks from `CRI` were attributed to `NET`, when the classifier was not trained on data from `CRI`. In the real-time scenario (E3), the NTP classifier attributed over 76% of the attacks in most cases, even outperforming the DNS classifier. Since NTP experiences less amplifier churn, booters can use the same amplifier set for a longer period of time, i.e., an attack is more likely to use a set of amplifiers for which the classifier already observed a training sample. A notable exception here is `BO1`, for which only 57% of the attacks could be attributed, despite the large number of attacks contained in the data set. This indicates that `BO1` performs rescans more frequently than other booters.

Averaging over booters for which the data set contains more than one attack, our classifier achieves a macro-averaged precision of 100.00% and recall of 86.25% in E1 for DNS, and 99.74% and respectively 91.01% for NTP. In the case of real-time attribution (E3), the precision stays similarly high (100.00% for DNS, 99.69% for NTP), while the recall drops to 69.35% and respectively 76.73%.

5.3 Attribution

After validating the classification mechanism, we now turn to applying it to our entire data set of attacks observed at the honeypots (excluding the self-attacks). Due to their low entropy, we excluded attacks that were only observed by a single honeypot. This left 266,324 NTP-based and 161,925 DNS-based attacks. For both we trained our classifier on all self-attacks collected from December 9 to February 10.

Our NTP classifier attributed 38,520 attacks (14.46%) to one of the booters it was previously trained on and our DNS classifier attributed almost a third of all attacks (49,297, 30.44%) to a booter. Note that not all attacks observed at the honeypots have to be caused by booters; they can also be caused by malevolent parties that do not offer their attack capabilities on an online platform. Furthermore, since we only trained our classifier on a limited set of booters, our classifier cannot possibly achieve a classification rate of 100%. Still, attributing a considerable amount of attacks to the booters of our training set indicates that the booters we considered are used very actively.

6 Victim-Driven Attack Attribution

Based on the success of the classifier that allows *honeypot operators* to attribute DDoS attacks, we now aim to build a similar classification method that will enable *victims* to attribute attacks based on features that can be extracted from victims' network traces. The core idea is to isolate a set of features that are directly observable by the victim and that can precisely attribute attacks to a particular booter service using a similar k-NN-classifier algorithm.

6.1 Description

Motivated by the fact that each booter abuses characteristic sets of amplifiers, we use the set of amplifiers as seen in the victim's attack traces as a feature for training our victim-driven classifier. However, the TTL value of the attack source used in the honeypot operator attribution technique is not directly observable by a victim, so we cannot use this feature in our victim based attribution method. The loss of the TTL value feature is mostly compensated for by the victim being able to see a larger set of amplifiers used by the booter service.

As we will show, this single feature is sufficient to build a classifier that can accurately attribute NTP, SSDP, and CharGen attacks from the victim's perspective. The one exception is that the set of open DNS resolvers used by individual booter services are less stable over time, likely due to churn. As a result, relying on the set of amplifiers as the sole feature for classifying DNS attacks will not provide the same classification performance as for the other three attack types. Therefore, we must identify additional entropy to improve the accuracy of our victim-based DNS attack classification technique. Based on our analysis of DNS attack traces captured at our victim server, we noticed that each booter service tends to send spoofed *ANY* requests for a very small number of mostly non-overlapping domain names. We thus complement the feature of amplifier sets with an additional feature over the set of domain names resolved in DNS attacks. That is, for DNS, the Jaccard index is computed both for the set of amplifiers and for the set of resolved domains, and the similarity score is the mean of the two computed Jaccard indices. For all other protocols (NTP, SSDP, and CharGen), we use the Jaccard index computed over the set of amplifiers.

In the victim-driven data set, all attacks are labeled with the booter service and we do not have any `unknown` attacks. However, we will evaluate the situation of unattributed attacks by performing the same E2 leave-one-out CV experiment as in Section 5.2. Given this, we select a cutoff threshold t to introduce a label for an `unknown` classification result that is used in the same way as in Section 5.2. We choose a conservative threshold of $t = 0.55$ for CharGen, $t = 0.60$ for DNS, $t = 0.55$ for NTP, and $t = 0.45$ for SSDP. In order to select the threshold value, the score of correct classifications and incorrect classifications were manually checked and a reasonably conservative value was selected for each attack type. Only attack instances in the training set for which the similarity score is no less than t were considered as potential neighbors. If no neighbor could be found for a test instance, it was classified as `unknown`.

6.2 Validation

To validate the results of our victim-driven classifier, we perform the same experiments as in Section 5.2. Table 4 shows the result of our victim-driven classifier experiments for DNS and NTP[11].

In E1, our DNS classifier achieved high attribution rates of 80% or more, except for `BO2`, `EXI`, `EXO`, and `VDO`, where a large fraction was also marked as `unknown`. However, in five cases the classifier also mistook attacks from one booter as coming from another. The higher number of false positives for DNS is attributable to the less stable set of DNS amplifiers abused by booters. These results are worse than those for the honeypot-driven classifier, possibly due to the fact that unlike organic sets of amplifiers, the honeypots do not churn over time. Misclassifications are even more prevalent in our E2 experiment, where in some cases the classifier confused over half of the attacks. While the number of misclassifications could be reduced by lowering the cutoff threshold, this would also cause a higher rate of `unknown` results in the other two experiments. Finally, in E3 the classifier shows similar performance compared to E1, with a slight degradation. However, this is expected, since if a booter service has just rescanned we will have no training samples that match the current set of amplifiers.

For NTP the victim-driven classifier generally performs better than for DNS. In the 10-fold CV (E1), the classifier correctly attributed 71% or more of the

[11] Results for CharGen and SSDP can be found in Section A.1.

Table 4: Victim-Driven Experimental Results

(a) DNS

		AUR	BAN	BO2	BO3	EXI	EXO	NET	RAW	SER	ST1	ST2	ST3	STA	SYN	THU	VDO
	samples	25	10	2	27	7	25	19	81	36	21	51	18	21	25	24	60
E1	correct	96	80	0	89	29	60	95	96	100	95	98	83	100	100	100	43
	unknown	4	20	100	4	71	36	0	4	0	5	2	11	0	0	0	52
	wrong				VDO 7		NET 4	EXO 5					SYN 6				BO3 5
E2	unknown	100	100	100	74	86	64	47	98	100	100	100	89	48	56	96	92
	wrong				THU 4 VDO 22	RAW 14	NET 12 STA 24	EXO 42 STA 11	SYN 2				SYN 11	EXO 43 NET 10	VDO 8 ST3 36	BO3 4	BO3 5 SYN 3
E3	correct	76	60	50	81	14	40	79	75	92	71	82	83	86	92	96	28
	unknown	24	40	50	19	86	56	5	22	8	29	18	17	0	4	4	72
	wrong						NET 4	EXO 16	SYN 1 EXI 1					EXO 14	ST3 4		

(b) NTP

		AUR	BAN	BO1	BO2	BO3	CRI	DOW	EXI	EXO	KST	NET	RAW	SER	ST1	ST3	ST4	STA	SYN	THU	VDO
	samples	23	15	40	3	28	27	27	7	61	20	29	82	15	21	19	27	22	28	22	61
E1	correct	100	100	95	0	100	100	96	71	97	100	86	100	100	100	89	100	91	100	95	100
	unknown	0	0	5	100	0	0	4	29	3	0	0	0	0	0	11	0	9	0	5	0
	wrong											EXO 10 CRI 3									
E2	unknown	100	100	100	100	100	74	100	100	34	100	10	100	100	100	100	93	100	93	100	100
	wrong						NET 26			NET 66		EXO 86 CRI 3					SYN 7		ST4 7		
E3	correct	87	73	90	0	96	89	81	43	90	80	69	98	87	86	74	93	77	86	86	97
	unknown	13	27	10	100	4	11	19	57	10	20	17	2	13	14	26	7	23	14	14	3
	wrong											EXO 10 CRI 3									

attacks for every booter, in many cases even more than 85%. As before, BO2 marks an exception due to the small number of attacks that were recorded for this booter. As already observed in the honeypot-based classifier, attacks from NET and CRI showed similar behavior. A third booter, EXO, that was only observed in the victim-based data set exhibits similar traits as well. While we were not able to verify that these booters are just different front ends of a multibranding booter, they account for almost all of the misattributions not only for NTP but also for CharGen. In E2 the classifier achieves a perfect result for most booters, with the exception of the previously mentioned group and two confusions between ST4 and SYN. Again, the results of our real-time classification experiment (E3) are as expected, with attribution rates of over 69% in all cases, except for EXI, whose recall drops from 71% to only 43%, due to the small number of attacks observed from this booter.

Overall, the victim-driven classifier achieves a macro-averaged precision of 91.65% and recall of 79.03% for DNS, while for NTP it performs better with 94.58% and respectively 91.07%.

7 Discussion

We now discuss potential ways to evade our attribution implementation and describe general limitations of our approach that we have not discussed so far.

7.1 Evasion

While our attribution methods have proven to work well as of now, they may be susceptible to evasion attempts by miscreants. A *mimicry* attacker could try to be attributed as someone else by following our methodology, i.e., learning the attack profile of another booter and copying its behavior. For example, she could use the same set of reflectors as the other booter for her own attacks. However, this involves a significant increase in terms of effort in comparison to Internet-wide scans. In addition, our TTL-based features are much harder to copy, as they encode the location of the booter service and are subject to changes for other booter locations. While such mimicry attacks are possible [2], given the complexity and overhead, we do not believe that attackers trying to trigger a *false* attribution constitute an actual risk in practice. For similar arguments, attackers that share lists of reflectors with each other would partially poison our analysis, but again TTL-based attribution may be safe against this. Our use of the set of domain names resolved as a feature for our victim-driven DNS classifier can be evaded by booter services selecting a larger pool of domain names that result in large replies and cycling through this pool.

An *evasive* attacker could try to evade our classification mechanisms. Attackers have full control over the traffic they generate, and thus could add noise. For example, one could randomize the set of reflectors used in the attacks, or spoof the initial TTL value within a range of possible values. It is unclear if a classifier could still keep up with such evasion attempts, but it may be possible to add additional features to enrich the classification, such as other characteristics (e.g., IP packet IDs, DNS transaction IDs), as those have shown characteristic patterns even if they were randomized [8]. In addition, honeypots that selectively respond to scan requests may survive such randomization [10]. Even if attackers randomize the set of reflectors, any subset will still be a subset of a unique mapping to a scanner. Lastly, randomizing the traffic does also incur a performance overhead to attackers, as they cannot reuse pre-generated packets.

Finally, attackers could try to map out the honey amplifiers using probing messages [3] if the honeypot amplifier data was made public for the DDoS service to use as an oracle. To avoid this evasion technique, access to the honeypot amplifier data is restricted to vetted entities, such as researchers and LEAs.

7.2 Limitations

Our in-the-wild experiments faced some limitations, as discussed in the following:

Honeypot Coverage: Regardless of our attempts to maximize the coverage of the honeypots, they missed significant fractions of the self-attacks, especially for SSDP and CharGen. This can be addressed by framing larger emulated responses to make the honeypots more attractive to attackers. The coverage for two of the main protocols, DNS and NTP, was significant, though, covering about 57% of the self-attacks. We therefore argue that our results are representative at least for these two protocols. In addition, there is no limitation of our methodology that would restrict its applicability to the two well-tested protocols.

Multi-Source Attribution: We assumed that attacks are caused by single sources (booters). If botnets launched amplification attacks, our features (e.g., TTL) would be unstable. To give an upper bound of attacks launched by botnets, we searched for attacks with several TTL values, as this—among other reasons—might be caused by distributed traffic sources. Less than 9.5% of attacks at the honeypots show more than 2 TTL values at a honeypot.

Other Attacks: Other types of DDoS attacks, such as SYN flooding or HTTP-based attacks, do not use reflectors and are thus not traceable with our proposed methods. Note that amplification attacks constitute the most common bandwidth exhaustion attack. This is also demonstrated by the fact that all booters advertise amplification attacks, while support for other attack types (e.g., HTTP-based attacks) is far less popular. To put things into perspective: we observed more than 8,900 amplification attacks per day.

8 Related Work

The general risk of amplification attacks was first illustrated in Paxon's seminal paper on reflection attacks [15] and then by Rossow's recent overview of amplification vulnerabilities in 14 UDP-based network protocols [18]. A wealth of further work analyzed amplification attacks, such as attempts to monitor and reduce the number of reflectors [1,4,11], analyses on detailed amplification vectors in specific protocols [4,12,24–26], studies on the impact of DDoS attacks [29], and proposals to detect and defend against amplification DDoS attacks [5,9,18,28].

Orthogonal to these studies, we investigated ways to perform *attribution* for amplification DDoS attacks. While concepts for closing the root cause of amplification attacks (IP spoofing) are well-known [14], little success has been made in *identifying* the spoofing sources. Our work thus constitutes an important element for law enforcement to identify and act upon information of booter services that are responsible for the majority of attacks. We follow a similar goal to IP traceback mechanisms [16,21–23,30], i.e., to find the source of "bad" (such as spoofed) traffic. While we also aim to reveal the source of the bad traffic, we focus on attack services rather than locating the networks that cause the traffic. In addition, the working principles behind the methods are inherently different. Most IP traceback methods are deterministic and can be guaranteed to find the correct source of traffic. However, at the same time, they impose requirements that are often not met in practice, such as that providers have to mark IP packets or collaborate to find traffic paths. In contrast, our proposed mechanism advances the field in that we do not require such a collaborative effort. In fact, despite being known for decades, automated traceback mechanisms have not been deployed by many providers. To tackle this problem, our approach merely requires a set of honeypots that anybody can set up, enabling a single party to perform attribution. On the other hand, our approach is limited to mapping amplification attacks to booter services, whereas traceback mechanisms could trace back any type of DoS traffic—down to the network that caused it.

Closely related to our work is AmpPot, as proposed by Krämer et al. [8]. This honeypot technology has enabled us to monitor thousands of DDoS attacks per day. We combine such data with observations of attack traffic emitted by booters, introducing the new concept of attributing amplification attacks to booters.

Our work was motivated by various research papers that shed light onto booter services using forensic analyses. Karami and McCoy were the first to monitor such booter services, studying the adversarial DDoS-As-a-Service concept [6] and observing that booters are a source for amplification attacks. Similarly, Santanna et al. analyze leaked databases and payment methods of 15 booters [19]. Related to our idea to fingerprint booters, Santanna et al. performed self-attacks of 14 booter services and also observed that the set of reflectors chosen by booters may have overlap across attacks [20]. We build upon this observation, find further correlations for attacks of booter services, and propose to use theses for attack attribution. Karami et al. [7] provide a detailed view on the subscribers and victims of three booters. They provide early attempts to map the infrastructures of booters, but do not perform any kind of attribution between attacks and booters or infrastructures.

Wang et al. [27] have studied the dynamics of attack sources of DDoS botnets, showing distinct patterns per botnet. While the authors provide first results that might enable them to predict future attack sources, they do not further investigate this matter. Our work is different in motivation and techniques in multiple respects. First, booters follow a completely different methodology than DDoS botnets, which rarely use IP spoofing. Second, we can leverage the observation that attackers scan for "attack sources" (amplifiers). Third, we perform attack attribution rather than prediction.

Recently, Krupp et al. [10] showed how to uncover the *scan infrastructures* behind amplification DDoS attacks, which in some cases could also be identified to be the attacking infrastructure. Although their work might seem similar to ours at first, there are key differences both in the goal and the methodology: While they use probabilistic reasoning to identify the scanners that provide the necessary reconnaissance for attacks, we use machine learning techniques to link attacks to the originating booters. Moreover, both approaches serve different demands: while their work aids in adding pressure on providers to cease illegal activities, our paper helps to generate forensic evidence that a particular booter has caused a specific attack, which can prove useful in prosecution.

9 Conclusion

Our work presented the first deep exploration of techniques for attributing amplification DDoS attacks to booter services. We present two precise attribution techniques based on carefully chosen features as part of a k-NN classifier. In order to evaluate the effectiveness of our techniques, we subscribed to a small set of booter services and launched self-attacks to collect a ground truth set of attack-to-booter-service mappings. We discuss the ethical framework used to collect this data set, which is similar to that of a previous study [7].

Our honeypot-driven technique attributes DNS and NTP attacks with a very high precision of over 99% while still achieving recall of over 69.35% in the most challenging real-time attribution scenario. Further analysis has revealed that 25.53% (49,297) of the observed DNS attacks can be attributed to just 7 booter services and 13.34% (38,520) of the NTP attacks can be attributed to 15 booter services. We have shared these findings with law enforcement agencies to help them prioritize legal actions against the wealth of booter services.

Our second technique extracts features out of a victim's network's traces and attributes attacks from the *victim's perspective*, which opens the possibility to offer a centralized DDoS attribution service. Using this technique, victims can learn the source of the attacks they face and could even compare two attacks to determine if they have been launched by the same actor (booter).

Acknowledgements This work was supported in part by the the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA) under grant 16KIS0656, by the European Union's Horizon 2020 research and innovation program under grant agreement No. 700176, by the US National Science Foundation under grant 1619620, and by a gift from Google. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

A Appendix

A.1 Additional Experimental Results

Table 5 shows our experimental results for victim-driven attribution for CharGen (precision 92.86%, recall 89.24%) and SSDP (precision 92.15%, recall 81.41%).

References

1. The Spoofer Project. `http://spoofer.cmand.org`.
2. M. Backes, T. Holz, C. Rossow, T. Rytilahti, M. Simeonovski, and B. Stock. On the Feasibility of TTL-based Filtering for DRDoS Mitigation. In *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses*, 2016.
3. J. Bethencourt, J. Franklin, and M. Vernon. Mapping Internet Sensors with Probe Response Attacks. In *Proceedings of the 14th Conference on USENIX Security Symposium*, 2005.
4. J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Proceedings of the Internet Measurement Conference 2014*. ACM, 2014.
5. Y. Gilad, M. Goberman, A. Herzberg, and M. Sudkovitch. CDN-on-Demand: An Affordable DDoS Defense via Untrusted Clouds. In *Proceedings of NDSS 2016*, 2016.

Table 5: Victim-Driven Experimental Results for CharGen and SSDP

(a) CharGen

		AUR	BAN	BO1	BO2	BO3	DOW	EXO	KST	NET	RAW	SER	ST1	ST4	SYN
	samples	26	3	41	2	27	27	60	21	18	78	42	20	26	26
E1	correct	96	0	93	100	100	100	97	76	89	99	100	100	100	100
	unknown	4	100	7	0	0	0	3	24	11	1	0	0	0	0
	wrong														
E2	unknown	100	100	100	100	100	63	48	100	39	100	76	100	100	100
	wrong						SER 37	NET 52		EXO 61		DOW 24			
E3	correct	88	33	80	50	96	85	93	57	78	96	86	90	92	88
	unknown	12	67	20	50	4	11	7	43	22	4	14	10	8	12
	wrong						SER 4								

(b) SSDP

		AUR	BAN	BO1	BO2	BO3	DOW	EXO	KST	NET	ST1	ST2	STA	VDO
	samples	20	17	40	2	27	28	60	21	28	17	17	25	49
E1	correct	95	76	95	0	100	96	98	95	100	18	88	96	100
	unknown	0	24	5	100	0	4	2	5	0	82	12	4	0
	wrong	VDO 5												
E2	unknown	95	100	100	100	100	100	100	100	100	100	100	100	90
	wrong	VDO 5												AUR 10
E3	correct	80	59	85	0	96	86	95	81	86	12	71	88	98
	unknown	20	41	15	100	4	14	5	19	14	88	29	12	0
	wrong													AUR 2

6. M. Karami and D. McCoy. Understanding the emerging threat of ddos-as-a-service. In *LEET*, 2013.
7. M. Karami, Y. Park, and D. McCoy. Stress Testing the Booters: Understanding and Undermining the Business of DDoS Services. In *World Wide Web Conference (WWW)*. ACM, 2016.
8. L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, and C. Rossow. AmpPot: Monitoring and Defending Against Amplification DDoS Attacks. In *Research in Attacks, Intrusions, and Defenses*. Springer, 2015.
9. C. Kreibich, A. Warfield, J. Crowcroft, S. Hand, and I. Pratt. Using Packet Symmetry to Curtail Malicious Traffic. In *Proceedings of the 4th Workshop on Hot Topics in Networks (Hotnets-VI)*, 2005.
10. J. Krupp, M. Backes, and C. Rossow. Identifying the Scan and Attack Infrastructures behind Amplification DDoS attacks. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS)*, 2016.
11. M. Kührer, T. Hupperich, C. Rossow, and T. Holz. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *Proceedings of the 23rd USENIX Security Symposium*, 2014.
12. M. Kührer, T. Hupperich, C. Rossow, and T. Holz. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks. In *Proceedings of the 8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.
13. A. Networks. Worldwide Infrastructure Security Report. `https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf`, 2015.
14. P. Ferguson, D. Senie. BCP 38 on Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. `http://tools.ietf.org/html/bcp38`, 2000.

15. V. Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. In *Computer Communication Review*, 2001.
16. A. Perrig, D. Song, and A. Yaar. StackPi: A New Defense Mechanism against IP Spoofing and DDoS Attacks. Technical report, 2003.
17. M. Prince. The DDoS That Almost Broke the Internet. `https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/`, 2013.
18. C. Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of NDSS 2014*, 2014.
19. J. Santanna, R. Durban, A. Sperotto, and A. Pras. Inside Booters: An Analysis on Operational Databases. In *14th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.
20. J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras. Booters - An Analysis of DDoS-As-a-Service Attacks. In *14th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.
21. S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *ACM SIGCOMM Computer Communication Review*, volume 30. ACM, 2000.
22. A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-Based IP Traceback. In *ACM SIGCOMM Computer Communication Review*, volume 31. ACM, 2001.
23. D. X. Song and A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, 2001.
24. X. Sun, R. Torres, and S. Rao. DDoS Attacks by Subverting Membership Management in P2P systems. In *Proceedings of the 3rd IEEE Workshop on Secure Network Protocols (NPSec)*, 2007.
25. X. Sun, R. Torres, and S. Rao. On the Feasibility of Exploiting P2P Systems to Launch DDoS Attacks. In *Journal of Peer-to-Peer Networking and Applications*, volume 3, 2010.
26. R. van Rijswijk-Deij, A. Sperotto, and A. Pras. DNSSEC and its potential for DDoS attacks - a comprehensive measurement study. In *Proceedings of the Internet Measurement Conference 2014*. ACM, 2014.
27. A. Wang, A. Mohaisen, W. Chang, and S. Chen. Capturing DDoS Attack Dynamics Behind the Scenes. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015.
28. X. Wang and M. K. Reiter. Mitigating Bandwidth-Exhaustion Attacks Using Congestion Puzzles. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, 2004.
29. A. Welzel, C. Rossow, and H. Bos. On Measuring the Impact of DDoS Botnets. In *Proceedings of the 7th European Workshop on Systems Security (EuroSec)*, 2014.
30. A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2003.