Lutu, A., Bagnulo, M., Pelsser, C., Maennel, O. & Cid-Sueiro, J. (2016). The BGP Visibility Toolkit: Detecting Anomalous Internet Routing Behavior. *IEEE/ACM Transactions on Networking, 24*(2), 1237–1250.

DOI: 10.1109/tnet.2015.2413838

The BGP Visibility Toolkit: Detecting Anomalous Internet Routing Behavior

Andra Lutu*†, Marcelo Bagnulo†, Cristel Pelsser‡, Olaf Maennel§ and Jesus Cid-Sueiro†
*Institute IMDEA Networks, Spain
†University Carlos III of Madrid, Spain
‡ Internet Initiative Japan
§Loughborough University, UK

***Abstract*—In this paper, we propose the BGP Visibility Toolkit, a system for detecting and analyzing anomalous behavior in the Internet. We show that *interdomain prefix visibility* can be used to single out cases of erroneous demeanors resulting from misconfiguration or bogus routing policies. The implementation of routing policies with BGP is a complicated process, involving fine-tuning operations and interactions with the policies of the other active ASes. Network operators might end up with faulty configurations or unintended routing policies that prevent the success of their strategies and impact their revenues. As part of the Visibility Toolkit, we propose the BGP Visibility Scanner, a tool which identifies limited visibility prefixes in the Internet. The tool enables operators to provide feedback on the expected visibility status of prefixes. We build a unique set of ground-truth prefixes qualified by their ASes as intended or unintended to have limited visibility. Using a machine learning algorithm, we train on this unique dataset an alarm system that separates with 95% accuracy the prefixes with unintended limited visibility. Hence, we find that visibility features are generally powerful to detect prefixes which are suffering from inadvertent effects of routing policies. Limited visibility could render a whole prefix globally unreachable. This points towards a serious problem, as limited reachability of a non-negligible set of prefixes undermines the global connectivity of the Internet. We thus verify the correlation between global visibility and global connectivity of prefixes.**

I. Introduction

The performance of the global routing system is vital to thousands of entities operating the Autonomous Systems (ASes) which make up the Internet. The Border Gateway Protocol (BGP) is currently responsible for the exchange of reachability information and the selection of paths according to specified routing policies. By tweaking the BGP configurations, the network operators are able to express their routing preferences, designed to accommodate myriad economic and technical goals. Despite the flexibility offered, the implementation of routing policies is a complicated process in itself, involving fine-tuning operations. Thus, it is an error-prone task and operators might end up with faulty configurations that impact the efficacy of their strategies or, more importantly, their revenues. Flawed routing policies cause for anomalies to emerge in the Internet, including interdomain prefix leaks, e.g., the case of Dodo leaking its full BGP routing table to provider Telstra in February 2012 [1], prefix hijacks, e.g., the well-known case of Pakistan Telecom hijack of YouTube [2] or prefixes not being distributed everywhere, e.g., the case where some multi-homed networks could not see the prefix of the DNS K root server [3]. Over the last years, a lot of effort has gone in the direction of identifying, classifying and eliminating some of these anomalies [4].

Withal, even when correctly defining legitimate routing policies, unforeseen interactions between ASes have been observed to cause important disruptions that affect the global routing system [5], [6]. The main reason behind this resides in the fact that the actual interdomain routing is the result of the interplay of many routing policies from ASes across the Internet, possibly bringing about a different outcome than the one expected. Consequently, in order to ensure the efficiency of their routing policies, ASes periodically control how their preferences resonate in the routing system using, among other, public looking-glasses or public BGP routing feeds.

In this paper, we argue that *prefix visibility* at the interdomain level is ~~an important~~ a flag useful to detect cases of faulty configurations or bogus routing policies, which disrupt the functionality of the routing system. We say that a prefix is *visible* to an AS if the latter has a stable active route in its BGP *Global Routing Table (GRT)* for the prefix in question. *We (loosely) define the* ***GRT*** *as the routing table provided by an Internet Service Provider (ISP) to its customers requesting a full routing feed.* Each ~~Internet Service Provider (ISP)~~ ISP maintains its own version of the GRT, which may vary from one network to another in terms of routes contained [7]. A prefix is *globally visible* when almost all the ASes in the Internet have an active stable route for it. In this paper, we show that the lack of global prefix visibility can offer early warning signs for anomalous events that, despite their impact, often remain hidden from state of the art tools, e.g., [8], [9]. Additionally, we show that such unintended Internet behavior not only degrades the efficacy of the routing policies implemented by operators, but can also point out problems in the global connectivity of prefixes. In order to evaluate the global visibility of a prefix, we compare the content of the GRTs from the ASes that make their routing data public. *We define* ***Limited-Visibility Prefix (LVP)*** *as stable long-lived Internet prefix that is visible in the GRTs of at least two different ASes and in at most 95%*[1] *of all the GRTs in the*

[1]The choice of the 95% visibility threshold allows for a 5% error in the routing tables sampling process, also accommodating possible glitches that may appear in the data.

*available sample. Contrariwise, we define the **High-Visibility Prefix (HVPs)** as the prefix that is propagated in at least 95% of all the GRTs in the available sample. Additionally, we identify the **Dark Prefixes (DPs)** [10], which denote the LVPs that are not covered by less-specific HVPs.*

The work we present in this paper focuses on developing techniques, tools and methodologies to assist network operators and researchers in understanding the manner in which BGP routing policies take effect in the Internet and which may be their possible impacts in the Internet ecosystem. The BGP Visibility Toolkit aims to detect and analyze anomalous interdomain behavior by separating the LVPs which are unintentionally generated in the Internet. The toolkit includes four different components, which are our main contributions:

1) We propose the **BGP Visibility Scanner**[2], a tool which allows network operators to check the visibility of their IPv4 and IPv6 prefixes and detect unintended policies. An earlier version of this tool is documented in [11]. We have publicly released the visibility scanner in November 2012. Ever since, the tool has been well received by the operational community[3]. The tool continues to evolve and to attract a large amount of attention and feedback [12], thus validating its usefulness for the operational community. In Section II, we provide the detailed description of the methodology we build into the scanner.
2) We assemble **a unique ground-truth dataset of 20,000 LVPs**, for which network operators themselves confirm which is the expected visibility status of the prefixes reported by the BGP Visibility Scanner. After comparing the expected visibility status with the actual one reported by the scanner, we label each of these LVPs as *intended* or, respectively, as *unintended* to have limited visibility. *An **unintended LVP** is a prefix whose visibility status in the BGP Visibility Scanner does not match with the intentions that the network operators reports in the ground-truth dataset. Contrariwise, the **intended LVP** is a prefix for which the intention of the network operator matches the visibility status we can observe with the Visibility Scanner.*
   Collecting feedback from operators regarding their strategies in terms of interdomain routing policies is not an easy task. We invite the users who query our tool to participate in a survey. The survey is optional and asks the users to provide more information regarding the observed visibility status of their prefixes. Additionally, we have actively been in contact with various operators who asked for our support while debugging their routing policies. The dataset brings additional value in that it accurately documents distinct causes for the limited visibility of prefixes in the Internet and provides a deep understanding of the routing conditions which allows them to emerge. It documents multiple cases of misconfiguration, unforeseen interactions and intentional routing policies effects. In Section III, we further expand on several of these examples.
3) We propose the **Winnowing Algorithm** [13], a machine learning classification algorithm able to automatically distinguish the *unintended LVPs*, caused by misconfiguration or unforeseen interactions between routing policies, from the *intended LVPs*, which emerge as a legitimate expressions of intentional routing policies in the Internet. Using the BGP Visibility Scanner as a data mining tool, we blend the per-prefix visibility information with the ground-truth information from active users of the tool to generate an alarm system for misconfiguration or bogus routing policies. The resulting Winnowing Algorithm has a 95% level of accuracy. This further proves that visibility features are generally powerful to detect anomalies ~~which~~that, despite their impact on the routing system, are hard to single out due to the limited and distributed nature of the data. We explain ~~the proposed approach~~ in more detail the proposed approach in Section IV.
4) We analyze the **correlation between visibility and reachability** of both IPv4 and IPv6 prefixes [14]. Faulty configurations, complex interdomain interactions or bogus routing policies not only impact the efficacy of the intended routing policies of ASes. Sometimes, ~~the impacted prefixes are prevented~~ these phenomena prevent the prefixes to be learned altogether, making the attached host globally unreachable. It is expected that *limited visibility does not necessarily imply limited reachability*, since ~~the~~ a less-specific high visibility covering prefix ~~provides~~ may provide reachability. This is no longer true for the DPs, which lack ~~a~~ the covering less-specific prefix with high visibility to ensure that the attached hosts are ~~still~~ globally reachable. We show that the lack of global visibility of a prefix does imply a certain risk on its global reachability, especially in the IPv6 Internet. While the IPv4 dark address space can be largely explained as route leaks or mistakes, this is not valid for the IPv6 DPs. We find that the subset of IPv6 dark prefixes ~~are~~ is highly unreachable. We believe that this is a serious problem for the IPv6 Internet, as limited reachability of a non-negligible set of prefixes undermines the global connectivity of the Internet. We expand on the methodology details and discuss our results in Section V.

II. The BGP Visibility Scanner

In this section, we describe the BGP Visibility Scanner, the tool we propose for identifying prefixes with limited visibility at the interdomain level from publicly available BGP routing data. The tool has been active and available for the operational community since November 2012, allowing any network operator to check the visibility status of their prefixes. At the time of writing, the visibility scanner detects on a daily basis more than 95,000 IPv4 and IPv6 *LVPs*. The daily set of prefixes with limited visibility can be further queried using the BGP Visibility Scanner public web-page. We collect feedback on the intended visibility status of the LVPs from ~~the~~ operators of the networks originating the prefixes and which are actively using our tool. This further enables us to verify if the intention

[2]The BGP Visibility Scanner is publicly available at **visibility.it.uc3m.es**

[3]We presented the BGP Visibility Scanner [11] in different network operators group meetings, including NANOG, LACNOG, UKNOF, EsNOG. We have also announced it on RIPE Labs [9].

of the origin network is reflected in the observed visibility status of its prefixes and to gather ground-truth on the various causes for LVPs.

A. The BGP Visibility Scanner Methodology

In Figure 1, we depict the main steps we follow for processing the raw data according to the BGP Visibility Scanner Methodology. The methodology is structured in three steps: First, we retrieve the raw BGP routing data. Second, we preprocess the raw data in order to obtain the GRTs. Third, we follow the Visibility Scanner Algorithm to evaluate the visibility of each prefix within the sample of available GRTs.

1) The Raw Data: Next, we expand on the first block of the methodology flow depicted in Figure 1. We collect the routing information from two major publicly available repositories at RouteViews and RIPE RIS. The two repositories gather BGP data throughout the world, at the time of writing deploying 24 different collection points, to which we further refer as *collectors*. The collectors periodically receive BGP routing table *snapshots*, i.e. the content of routing tables at a certain moment in time from one or more routers within the ASes ~~activated~~ active as monitors. A *monitor* represents a network (identified by a unique Autonomous System number) that connects to the public RIS/RouteViews repository to regularly propagate its routing table snapshot. At the time of writing, there are more than 350 different active monitors in RIS and Routeviews. The monitors have different policies with respect to the public repositories, thus providing different types of routing table snapshots. We are able to identify three different types of feeds that the collectors receive, namely *Partial Routing Tables*[4], *Global Routing Tables* and *Global Routing Table with internal routes*[5]. However, only by comparing GRTs, we can identify the *HVPs* and *LVPs*. We further explain next how we differentiate between these three types of routing table feeds in order to evaluate the visibility of the prefixes in the GRTs alone.

Additionally, in order to address the confusion that converging prefixes generate in our analysis by emerging as false positive LVPs, we analyze two 8-hours apart samples of raw routing data. In other words, we retrieve the BGP routing tables twice every day, namely at 08h00 and 16h00 UTC. We use the two different samples in order to ensure the correct separation of routes with limited visibility as long-term expressions of routing policies implemented by ASes and converging routes or routes with temporary limited visibility due to internal operational activities of the ASes.

2) GRTs: Cleaning and Pre-processing the Raw Data: We explain in detail the steps we take in order to pre-process the raw data, as depicted in the second block in Figure 1.

Size Filter: in order to exclude the Partial Routing Tables from the whole set of feeds, we verify the actual size of the routing table. We consider that an IPv4 GRT should have no less than 400,000 routing entries [15]. Similarly, an IPv6 global routing table should not contain less than 10,000 routing entries. Consequently, we keep for further study only the routing feeds that comply with the minimum limit on the number of prefixes.

Clean GRTs: we perform a couple of *sanitary* checks on the actual data contained in the identified GRTs, in order to further discard the information that is of no interest for our analysis, namely bogons, MOAS and internal routes. **Bogons** are defined as *Martians*, representing private and reserved address space or *Fullbogons*, which include the IP space that has been allocated to a Regional Internet Registry (RIR), but has not been assigned by that RIR to an actual Internet Service Provider (ISP) or other end-user. Using the bogons lists published for IPv4 and IPv6 on a daily basis at [16], we build the *bogon filter*. We apply this filter on the content of all the GRTs, to eliminate any matching or more-specific prefixes for the ones present in the bogon list. The *Multiple-Originating AS (MOAS)* [17] prefixes cannot be qualified within our study, since for these prefixes we are not able to identify which origin AS might be suffering/generating the reduced visibility of its prefixes. We eliminate all the identified MOAS. We filter out the cases of prefixes emerging as *LVPs* that are, in fact, ~~are~~ **internal routes** propagated to the collectors by the monitors. To discard any potential internal paths, we remove all the prefixes visible to only one monitor, which is also the origin AS for the prefixes in question.

3) The Visibility Scanner Algorithm: In this section, we expand on the details of the proposed Visibility Scanner Algorithm. Having obtained the *"clean"* version of the GRTs, we identify in this phase the prefixes with *stable* limited visibility in the Internet. When deriving the final visibility label, we account for the dynamics of a prefix in time, as ~~presented~~ we show in the last block from Figure 1.

Labeling Mechanism: we use the two different GRTs taken 8-hours apart (i.e. 08h00 and 16h00 UTC) in order to ensure the correct separation of the external long-term expressions of the routing policies ~~implemented by~~ an individual AS implements and the converging prefixes or the temporary LVPs due to internal operational activities. We evaluate the *visibility degree* at both sampling moments and assign a *visibility label* at each time. We define the *visibility degree* as the number of GRTs within the daily sample which contain (i.e., "see") a certain prefix. The *visibility label* shows the visibility status of each prefix at the sampling time, i.e. *LVP* for Limited Visibility Prefix and *HVP* for High Visibility Prefix. We evaluate the *visibility degree* of all prefixes at the two different sampling moments and assign a *visibility label* at each time. We label ~~LVP~~ with *LVP* the prefixes present in *less* than 95% of the GRTs at each sampling time. Otherwise, ~~the prefixes are labeled HVP~~we label the prefixes with *HVP*. Consequently, we assign to each prefix at most two *visibility labels*.

Label Prevalence Sieve: at this point, we identify and discard LVPs caused by other factors than routing policies, e.g. BGP convergence. By analyzing the two visibility labels that we assign in the previous phase to every prefix, we aim to avoid such false positive LVPs. The high visibility of a

[4]This type of feed can be though as the result of establishing a peering-like business relationship between the monitor and the collector. By definition, these feeds are not GRTs, thus are not useful for our analysis.

[5]In some cases, it may happen that the monitor announces, aside from the complete routing table, other additional internal information. This additional information is again of no interest for our study, since we do not focus on the internal operations of a network. Consequently, we need to identify and filter out these particular routes within the complete routing feed.

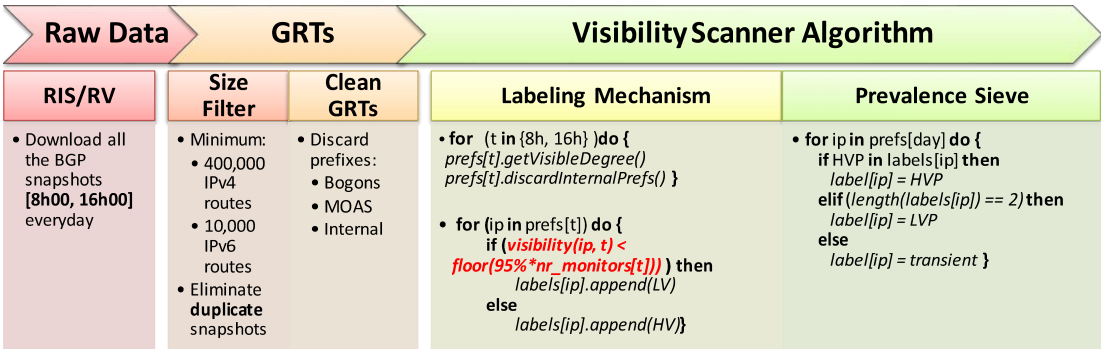

Fig. 1. The BGP Visibility Scanner methodology.

TABLE I
IPV4 AND IPV6 BGP ROUTING DATA PROCESSING STATS.

Data Stats	IPv4	IPv6
Number of GRTs	150	110
GRT size filter	400,000	10,000
Total Prefixes Analyzed	550,000	16,500
Number of LVPs	90,000 [20%]	3,500 [20%]
Number of HVPs	420,000	12,500
Number of DPs	2,500 [3%]	500 **[14%]**
% ASes originating LVPs	9%	13%
Internal Routes	10,000	150
Converging Routes	8,000	10

prefix at either of the two sampling times hints the fact that the route can be visible to all ASes active as monitors. Should this change during the analyzed time, it might be a cause of, for example, topology changes or failures. Therefore, we consider that *the HVP label always prevails*, i.e. if a prefix is tagged as *HVP* at one sampling time, it receives a final label of *HVP*. Otherwise, when the prefix has no *HVP* label, we analyze the cases of ~~*LVP*~~*LVPs* ~~prefixes~~ emerging in our results. If a prefix appears only at one sampling time and it has ~~a~~ an *LVP* label, this might be a sign that the prefix is in the process of being withdrawn or, contrariwise, in the process of converging after just being injected. These particular routes cannot be qualified within the visibility scanner, thus we discard any prefix with only one visibility label and that label being *LVP*. The only case where we can say a prefix has final limited visibility is when it has exactly two *LVP* labels.

B. The LVPs in Rough Numbers

~~The~~ We make publicly available the set of LVPs identified using the BGP Visibility Scanner methodology~~is made publicly available~~, so that each network can potentially check the status of its prefixes. ~~The results are refreshed~~ We refresh the results on a daily basis~~such that the operators can have~~, to give operators an updated view on the efficiency of their routing policies, both in IPv4 and IPv6. We present next a few statistics regarding the number of both IPv4 and IPv6 LVPs ~~, as observed~~ that we observe during the first 15 months when the BGP Visibility Scanner has been active (i.e., from November 2012 until January 2014)~~when the BGP Visibility Scanner has been active~~. We summarize this information in Table I.

Every day we collect more than *500* routing feeds, for each of the two different sampling moments. In rough numbers, the daily total number of IPv4 prefixes is around *550,000 prefixes* at the time of writing. Out of these, around *10,000 IPv4 prefixes* are internal routes, which we discard. We also remove the converging routes. This incurs the elimination of about *8,000* additional IPv4 prefixes in average. For the remaining prefixes, we continue the visibility analysis and assign LVP/HVP labels. We identify, in average, *90,000 LVPs* and *420,000 HVPs*. When checking how the two sets of prefixes overlap, we find that there are more than *2,500* IPv4 LVPs without covering HVP, which we mark *DPs*. We ~~have observed~~ observe that more than 3,800 ASes ~~which~~ inject LVPs, out of which less than 1,000 ASes originate DPs.

The size of the IPv6 GRT is much smaller than the one for IPv4. We adjust the size filter to be at least 10,000 routes for IPv6. We identify and further process the content of 110 GRTs to determine the set of IPv6 LVPs. The daily overall total number of prefixes is approximatively *16,500 prefixes*. Out of these, ~~on~~ we discard in average *150 IPv6 prefixes* ~~are discarded as~~ that are internal routes advertised to the collector. We further eliminate the converging routes, i.e., approximatively *10* additional prefixes in average. On average, *3,500 IPv6 prefixes* are *LVPs* and approximatively *12,500* IPv6 prefixes are *HVPs*. In other words, 20% of all the IPv6 prefixes identified from all the analyzed routing tables are LVPs. This is consistent with the result for the IPv4 LVPs, where out of all the prefixes learned, 20% have limited visibility [11]. When checking how the LVPs and HVPs overlap, we find that, for IPv6, there are more than *500* LVPs without a covering HVP, which we label as *DPs*. This represents approximatively 14% of the whole set of IPv6 LVPs. When comparing with the situation in IPv4, where in average only 3% of the LVPs are ~~marked as~~ *DPs*, we find that we have almost 5 times more IPv6 dark address space. This is relevant because these prefixes may have limited reachability, in the lack of a default route. We further analyze the correlation between the limited visibility and the limited reachability of the LVPs in Section V. We observe that more than 13% of all IPv6 active ASes inject LVPs and approximatively 5% of all IPv6 active ASes originate DPs. In IPv4, we see that 9% of all ASes originate LVPs and only 2% are also injecting DPs. These numbers may vary from day to day, given that neither the monitors providing their global routing tables, nor the actual content of the GRTs are constant over time.

III. GROUND-TRUTH: UNDERSTANDING *LVPs* THROUGH OPERATIONAL USE CASES

The daily set of LVPs is accessible in the BGP Visibility Scanner for queries on a per-origin AS basis. At the time of writing, the tool gathers over 8,000 queries for more than 3,000 different origin ASes. We invite the users of the tool to

participate in a survey regarding the expected visibility status for the prefixes retrieved. We also inquire about the possible causes generating the LVPs that the scanner detects. By doing this, we aim to obtain more insights into why LVPs appear in the Internet. Also, network operators directly contact us to provide information on the intended visibility status for their *LVPs* detected with the visibility scanner. Leveraging the feedback received, we build a unique ground-truth dataset including 20,000 *LVPs*. We compare the original intention of the operator with the actual observed visibility status of the prefixes in the BGP Visibility Scanner, and distinguish two classes of *LVPs*: ***intended*** and ***unintended***. The ground-truth dataset contains 1,150 prefixes of the class *intended* and a staggering 18,850 *LVPs* of the class ***unintended***. Consequently, the dataset documents a significant variety of factors which generate LVPs, both intentionally and unintentionally. We expand next on a few operational examples which we ~~found~~ find to be most interesting.

A. Intended LVPs

Some ASes create *LVPs* on purpose. There are several ways this can be done, including the use of BGP communities to restrict the scope of a prefix advertisement (e.g. geographically restricted prefixes aimed to offer connectivity only to networks located in a certain region) or advertisements only through (some) peering and not transit ~~relationships~~links. We next provide real cases of ASes that deliberately restrict the global propagation of their prefixes.

For example, using the BGP Visibility Scanner, we are able to verify and validate the routing policies of two of the Internet DNS root-servers. For each root-server we identify the presence of one more-specific *LV* prefix, which is meant ~~for providing~~ to provide connectivity only to direct peers and, consequently, is tagged with the well-known *NO-EXPORT* community. The limited visibility of the more-specific prefix correctly reflects the impact of the *NO-EXPORT* community on the connectivity of the prefix. However, the *LVP* has global reachability due to the presence of a less-specific *HVP*, which is used by the root-servers in order to avoid connectivity issues.

The tool also validates the routing policy of a large content provider that deliberately limits the visibility of one of its prefixes to a certain geographical area.

B. Unintended LVPs

The second type of use cases we present captures unintended results of routing policies, i.e., accidental misconfiguration or unforeseen interactions between external routing policies at the interdomain level.

In many cases, *LVPs* are the result of errors in the configuration of filters at the origin or other ASes that have received the prefix announcement. For example, a large and widely-spread ISP learned that a large set of its internal prefixes are visible only to some of its direct peers. After further investigation, the ISP was able to identify the misconfiguration of its outbound filters, which should have otherwise ensured that those prefixes were not being advertised to other networks. After correcting these issues, the origin AS successfully eliminated

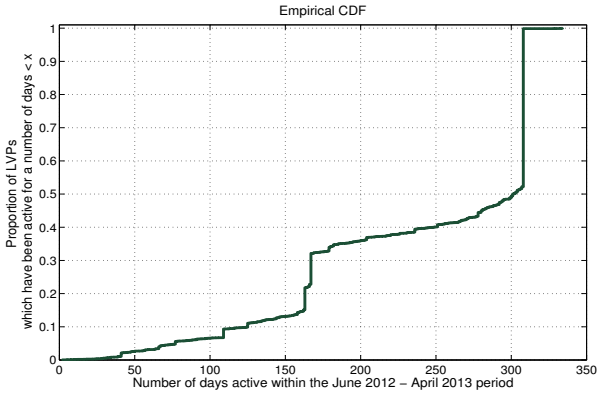

Fig. 2. Empirical CDF of the *unintended LVPs* lifetime, from June 2012 until the end of April 2013.

4,000 unintended LVPs of whose existence it was previously unaware. We note that these issues ~~were~~ remained undetected for a very long time before the BGP Visibility Scanner became operational. Figure 2 further depicts the empirical CDF of the time unintended LVPs were active within the 11 months period between June 2012 until the April 2013. We observe a large number of *LVPs* were active for 150 days within those 11 months. These LVPs then disappeared around November 2012, after the origin ASes learned that it was accidentally leaking 4,000 prefixes because of misconfiguration.

A clear example of the serious impact that undetected mistakes might have on the strategies of a network is the case of an ISP whose prefixes appear as *DPs* in the scanner. After investigating this issue, the origin AS found that, due to a mistake in the configurations of its transit provider, the ~~prefixes are not being correctly advertised~~ISP was not advertising the prefixes of the AS in question. This not only means that the prefixes are not globally propagated, but that they could also be suffering from limited reachability in the Internet.

Unforeseen interactions between legitimate and correctly defined routing policies of ASes can also limit the visibility of some prefixes at the interdomain level. The ground-truth dataset we collect includes cases of ASes ~~reporting~~ which report that the limited visibility of their prefixes is due to the impact of the filtering policies of third-party ASes. More exactly, ~~the *LVPs* the scanner detects~~ these are cases of prefixes that do not have corresponding objects defined in the Regional Internet Registry (RIR) database. Thus, ASes that filter based on the information from the RIR database discard such prefixes, impacting their global visibility. This, consequently, causes prefixes to suffer from unintended limited visibility. Using the BGP Visibility Scanner, network operators are able to discover and address such cases of unintended LVPs.

Overall, as far as we know, the BGP Visibility Scanner has been able to eliminate approximatively 18,500 unintended LVPs withing the first 15 months of activity. We find that 14,600 out the total 18,500 known unintended *LVPs* were already active on June 1st 2012, as much as 5 months before the BGP Visibility Scanner became available.

IV. Winnowing Unintended LVPs: the Machine Learning Approach

In this section, we propose the Winnowing Algorithm, the machine learning tool we incorporate in the BGP Visibility

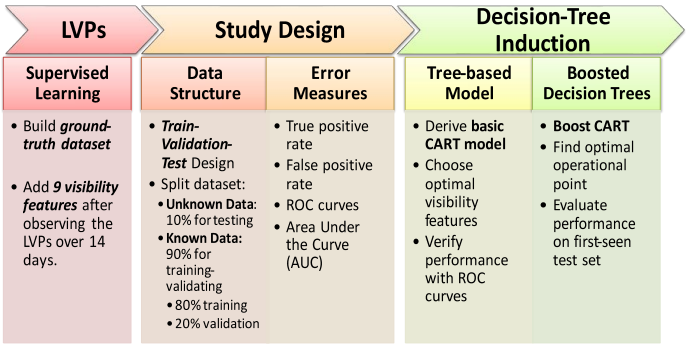

Fig. 3. Winnowing Unintended LVPs: detailed methodology.

Toolkit to automatically separate ***unintended LVPs***, which emerge because of errors or complex interaction between networks from the ***intended LVPs***, which emerge as expected expressions of routing policies. The tool builds on the ground-truth dataset collected with the BGP Visibility Scanner, which we present in Section III. We use the BGP Visibility Scanner as a a *data mining tool* for identifying stable LVPs and monitoring their status in time. The supervised learning approach advances a decision tree model using specific visibility features in order to classify the *LVPs*. We further show that the per-prefix visibility features derived by monitoring the prefix visibility status reported by the visibility scanner over a period of two weeks are generally powerful to detect prefixes which are suffering from limited unintended visibility.

Figure 3 illustrates the different steps we follow to winnowing unintended LVPs from the rest. We begin by describing the visibility features we analyze to characterize each prefix from the ground-truth dataset of 20,000 pre-classified LVPs. We then present the proposed machine learning study design and expand on the error measures we try to optimize. We advance a decision tree model using the optimal set of visibility features, chosen according to the *information gain* measure. Using the AdaBoost [18] algorithm, we boost the obtained basic model to achieve higher accuracy and reduce bias in this supervised learning approach. We finally test the boosted tree-based model on a hold-out dataset, which was not used during the learning phase.

A. Data for Supervised Learning

We expand here on the way we pre-process the ground-truth dataset to further use it for supervised learning. This corresponds to the first step we illustrate in the ~~workflow~~ work-flow in Figure 3. The ground-truth consists of 20,000 *LVPs*, each pre-classified to indicate if the prefix has ***unintended*** limited visibility or if it is the consequence of ***intended*** interdomain behavior. We note that the dataset exhibits an important disproportion between the two defined classes, with 1,150 prefixes of the class *intended* and 18,850 *LVPs* of the class *unintended*. In order to use the ground-truth for training the machine learning Winnowing Algorithm, we first identify the full set of significant visibility features, which we attach to each prefix in the dataset. For every *LVP*, the corresponding origin AS is observed over a period of *14 days* prior to the feedback moment. We can thus characterize the visibility dynamics captured in the BGP Visibility Scanner. All the possible visibility parameters are listed and explained in Table II.

B. Study Design

In this section, we explain the study design we follow for deriving the Winnowing Algorithm. This is depicted in the second processing block in Figure 3.

We design the learning process in a training-validation-test format. In other words, we use cross-validation to estimate how the classification model behaves on an independent never-before-seen set of data. Also known as *rotation estimation*, this approach implies splitting the data into *known data*, which we use for training and validation, and *unknown data*, also known as *hold-out test data*, which we use for final testing. The idea of cross-validation is to repetitively split the known data into training and validation disjoint sub-sets, in order to estimate the accuracy of the model. We use the training dataset to first derive the classification algorithm. In order to avoid issues like over-fitting and gain more insight on how the model could generalize to new independent data, we then perform an initial testing on the validation data. We further tune the decision model to achieve optimal performance on the validation dataset. We manually repeat the training and validation for various ~~splits of~~ ways to split the known data. In order to determine which is the algorithm with the best results across all possible data splits, we define a set of error measures ~~which we further~~ that we explain in detail next.

1) Data Structure: When splitting the ground-truth dataset into training, validation and the hold-out test data, we need to fulfill several constraints, which we explain next. The three datasets considered in the study design must be perfectly disjoint, i.e., we should observe no prefixes nor origin ASes in any two or more datasets. We thus split the ground-truth such that all the LVPs generated by the same AS are included in one unique dataset. We impose these restrictions in order to ensure a correct estimation of the algorithm performance when predicting the class of LVPs originated by ***unknown ASes***, on which we have no prior ground-truth for training. This is a challenge, since predicting the class of LVPs originated by an AS on which we have previously trained is significantly easier.

We first create the hold-out test dataset, by randomly choosing 10% of all the ASes ~~which~~ that provided feedback. This hold-out test set is under no circumstances to be used in the training-validation phase of the learning process. Its main purpose is to estimate the performance of the optimal winnowing algorithm on unknown independent data.

We split the remaining 90% of the ground-truth data in two different subsets, namely the training and validation datasets. We perform the separation such that the training dataset has approximatively 80% of the remaining ground-truth dataset, and the validation set, the 20% left. We ~~require that this constraint is respected~~ respect this constraint for the total number of prefixes and also for the number of different ASes, i.e., 80% of ASes must be in the training dataset and the rest of 20% in the validation dataset. Additionally, we require that the 80-20 split for the training-validation datasets is also respected for each of the two classes of prefixes. In other

TABLE II
THE LIST OF PER-LVP VISIBILITY FEATURES. ALL THE VALUES ARE CALCULATED FOR AN OBSERVATION PERIOD OF **14 DAYS**. THE FEATURES ARE ORDERED IN DECREASING ORDER OF THEIR IMPORTANCE, ACCORDING TO THE INFORMATION GAIN.

Extracted per-LVP Feature	Explanation	Information Gain [weights]
`mean_nrPrefs`	Average number of LVPs generated by the same origin AS	0.319
`mean_MonitorsDetecting`	Average *proportion of active monitors* detecting the LVP	0.308
`std_MonitorsDetecting`	Standard deviation of the *proportion of active monitors* detecting the LVP	0.3068
`std_nrPrefs`	Standard deviation of the number of LVPs generated by the same origin AS	0.3060
`mean_VisibilityDegree`	Average *absolute visibility degree* for the LVP	0.244
`std_VisibilityDegree`	Standard deviation of the *absolute visibility degree* for the LVP	0.234
`length`	Prefix length of the LVP detected by the BGP Visibility Scanner	0.183
`TimeActive`	Proportion of time the LVP remained with limited visibility	0.153
`VisibilityLabel`	Shows if the prefix has a covering less-specific HVP (i.e., LVP) or not (i.e., DP).	8.61e-05

words, we must have 80% of the intended LVPs in the training and the rest 20% in the validation dataset and the same for the unintended LVPs. We impose these rules to ensure a similar distribution of prefixes and ASes in the training and in the validation datasets. Given ~~the explained~~ these constraints, we identify exactly 989 different ways in which we can separate the training-validation datasets. In rough numbers, this means training on about 15,000 *LVPs*, validating on about 4,500 *LVPs* and, finally, testing on approximatively 100 *LVPs* which were not used in the training-validation process.

2) Error Measures: In this section, we define the error measures which we choose to describe the performance of the derived classification algorithm. The *accuracy* of a classifier is defined as the percentage of ground-truth tuples which are correctly classified when we test on a set of unknown data. However, even when we obtain a very high value for the accuracy of the classifier, it may be the case that the model does not recognize very well the tuples of one of the two classes~~,~~. This happens especially when dealing with *unbalanced* classes in the data, which is our case. To address these limitations and to further evaluate the model performance, we define the following concepts:

- *True Positive tuples [TP]*: number of tuples classified as unintended, which really are of unintended class.
- *False Positive tuples [FP]*: number of tuples classified as unintended, which really are of intended class.
- *True Negative tuples [TN]*: number of tuples classified as intended, which really are of intended class.
- *False Negative tuples [FN]*: number of tuples classified as intended, which really are of unintended class.

We now can define the two error metrics which allow us to correctly evaluate the performance of the classification by capturing the per-class classification accuracy. Namely, we use **True Positive rate** [TP_{rate}] and **False Positive rate** [FP_{rate}], which we calculate as follows:

$$TP_{rate} = P(unintended \mid unintended) \sim \frac{TP}{TP + FN},$$
$$FP_{rate} = P(unintended \mid intended) \sim \frac{FP}{TN + FP}.$$

In other words, the TP_{rate} represents the probability of predicting a tuple[6] as unintended, conditioned by the fact that the tuple is indeed unintended. Similarly, the FP_{rate} represents the probability of classifying a tuple as unintended, conditioned by the fact that the tuple is actually intended.

[6]We call *tuple* the data structure consisting of the LVP and the corresponding visibility features.

~~We use the *Receiver Operating Characteristic (ROC) curves* to visualize the performance of a classifier . The ROC curve is a graphical plot that illustrates the performance of a classification model as~~

For each tuple, the decision tree classifier computes some value between zero and one that can be interpreted as an estimation of the probability that the tuple belongs to the ~~discrimination threshold is varied. Many classification models, including decision trees, assign a probability to every tuple, expressing the degree to which the tuple is considered to belong to one of the positive class(i. e., in our case the LVP class "unintended"). By setting a discrimination threshold~~"unintended" class. The categorical decisions of the classifier are the result of setting a threshold, μ, on these probabilities, ~~we obtain a categorical classifier, i.e.,~~ in such a way that the tuples are classified as ~~*unintended*~~ "unintended" if their probability is higher than ~~the fixed threshold, and "intended" otherwise. The performance of such a model is characterized by a single (TP_{rate},~~ μ, and as "intended" otherwise. It turns out that the TP_{rate} and the FP_{rate} depend μ. For large μ only the most probably tuples are assigned to the positive class, which is useful to minimize false positives (i.e., small FP_{rate})~~pair of values which can be plotted in the so-called ROC space.When considering different values of the discrimination threshold, we obtain a set of points capturing the~~ , but at the expense of missing tuples from the positive class (i.e. small TPrate). On the opposite, smaller thresholds increase TP_{rate} ~~to FP_{rate}~~ , but at the expense of increasing FP_{rate}. The two dimensional plot representing pairs (FP_{rate},TP_{rate}) for different values of μ is usually referred as the Receiver Operating Characteristic (ROC) curve. The ROC curve is a standard way of representing the trade-off ~~which can be plotted in the ROC space. Together,these points form the ROC curve of the decision model~~between the positive and the negative class in binary classification. Overall, the ROC curve gives an aggregated view on the performance of the model, without reference to a specific threshold value.

To ~~asses~~ assess the general performance of various models using the ROC space, we ~~can~~ measure the area under the curve (AUC). The ROC space usually shows an ascending diagonal line, corresponding to the ROC curve of a non-informative classifier (i.e. one making stochastic decisions independent on data). As the ROC curve goes closer to this line, the AUC goes closer to 0.5 and the model becomes less accurate, up to the point of random. Consequently, an AUC closer to 1 shows high performance for the model and an AUC close to 0.5 shows low performance.

We ~~also~~ use the ROC curve to further tune the model and determine the optimal operating point for the classification model. Note that, since each point in the curve corresponds to a different value of the discrimination threshold, selecting the operating point is equivalent to selecting a discrimination threshold. This selection may depend on the design considerations. For instance, if positive and negative examples are equally likely, the operating point maximizing the sum between the TP_{rate} and $1 - \text{FP}_{\text{rate}}$ could be a good choice, because this is equivalent to maximizing the number of correctly detected tuples. However, the decision model operating with this threshold may not provide good results on new datasets with different distributions of tuples per class. To this end, a robust choice of the operating point is the *break even point*. The latter represents the value of the discrimination threshold where FP is equal to FN. It has been proven that this point optimizes the performance of the classifier under worst case conditions, i.e. under adversarial choices of the class distributions [19].

C. Decision Tree Induction

After previously defining the data structure, error measures and tools for assessing the performance of a classification model, we next explain the constructive process we follow in order to derive the decision tree-based Winnowing Algorithm. Following the flowchart depicted in Figure 3, we thus proceed to the last block, namely the Decision-Tree Induction.

In the model, we choose decision trees as base learners which are boosted to create a robust classification model. Tree-based learning methods rely on iteratively partitioning the data into smaller groups of similar elements [19]. The splitting of the data is done using the features that best separate the two classes, *intended or unintended* LVPs. The key idea is to chose the splits which maximize the group homogeneity, i.e., elements of the same class are within the same group, or until the groups are sufficiently "pure". Choosing the right number of splits is a challenge~~, since we~~. We can easily overfit the model by considering splits that are very specific to the training data~~, or, contrariwise,~~. Contrariwise, we can underfit it by considering shallow general splits. Finding the correct balance is conditioned by finding the optimal set of features used to partition the data.

Decision tree induction is the process of deriving decision trees from the training ground-truth datasets [19]. We use the extensively tested and popular machine learning method called Classification and Regression Trees (CART) [20] for deriving and fine-tuning the base tree model. A CART tree is a binary decision tree that is constructed by splitting the training dataset into subsets based on a feature value test. The process is then repeated on each derived subset in a recursive manner. The recursion is completed when all the elements in a subset at a node are of the same class or when splitting no longer adds value to the classification. We derive the decision trees using the standard library *tree* for the R Project for Statistical Computing [21]. Using each of the 989 different training-validation splits, we determine the optimal decision tree in every case. The resulting optimal CART trees are further used in the following step, where by boosting we combine multiple CART base learners to form a robust classification model.

In Algorithm 1, we show the main phases ~~traversed~~ we traverse in the process of Decision-Tree Induction~~that~~, which results in building the final Winnowing Algorithm.

Algorithm 1 Decision-Tree Induction

1) **Feature Selection**
   - for $n = 1, ..., N$:
   - **Learning from n features:**
     - append the n-th feature (ranked using the information gain) to LVPs in the training set;
     - for each of the 989 train/validation splits:
       - grow CART base model;
       - for each discrimination threshold value:
         - compute TP_{train}, TN_{train}
     - compute **average ROC** over the 989 ROC curves;
     - compute AUC(n) for the averaged ROC curve;
   - take subset of n* features maximizing AUC(n).
2) **Boosting**
   - for each train/validation split (out of 989 possible) :
     - train AdaBoost using n* features
     - compute TP_{train}, TN_{train}
   - compute threshold average ROC curve over all the splits;
   - take threshold value from the break-even point in the average ROC.
3) **Testing**
   - train AdaBoost using n* features and the whole dataset, excluding hold-out test data;
   - compute TP, TN on the test set.

1) Feature Selection: We expand next on the Feature Selection phase of the Decision Tree Induction process, ~~succinctly described~~ which we succinctly describe in Algorithm 1. We perform the feature selection by ordering the visibility features in ~~decreasing order~~ decreasingly, in function of their *information gain*. The information gain is a widely accepted measure for evaluating the capacity of a feature to distinguish between tuples of different classes. In the third column of Table II, we show the values of the information gain metric associated to each of the 9 different visibility parameters. In order to select the subset of features which ensures the optimal performance of the base CART decision tree for any training-validation data configuration, we adopt a progressive approach.

For all the LVPs in the known training dataset, we begin by considering ~~that every tuple contains~~ the tuple formed by the LVP and ~~only~~ the feature with the highest value of the information gain, i.e., the *mean_nrPrefs*. For each of the 989 training-validation splits of the known data, we then grow ~~a~~ the decision tree that uses only the value of the *mean_nrPrefs* feature to discriminate between the two classes of LVPs. We then test each of these 989 decision trees on the validation dataset ~~of the corresponding data split~~ and derive one ROC

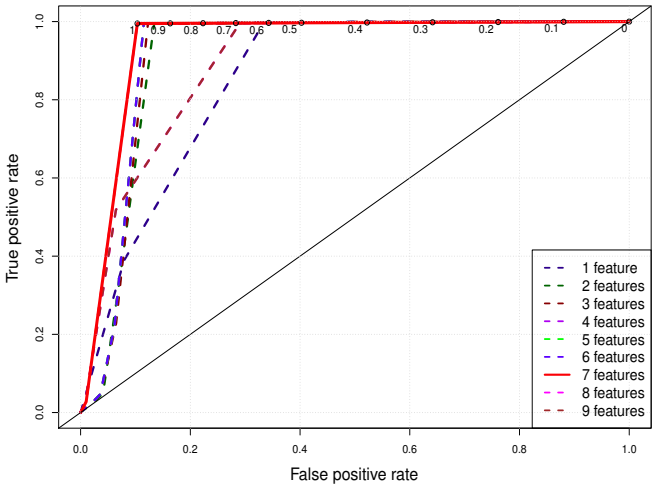

Fig. 4. Threshold-average ROC curves for performance estimation of the decision tree built with the 9 feature-sets. The red continuous curve for the model using the 7 most important features has the highest AUC and, thus, constitutes the optimal model.

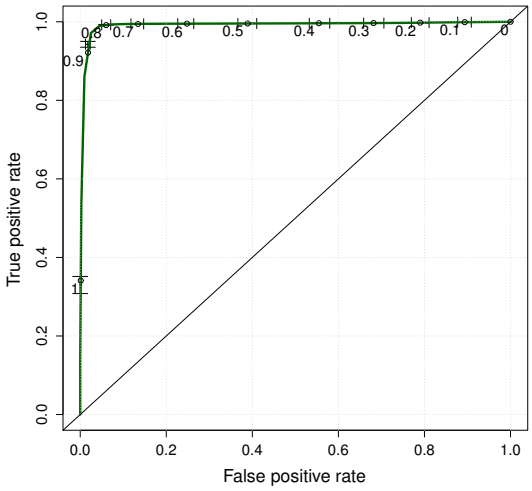

Fig. 5. Threshold-average ROC curve of the boosted decision trees derived using each of the 989 possible data splits.

curve for each of the 989 decision trees. We then calculate the *average* performance ~~of the~~ over the 989 decision trees generated for each of ~~these~~ the 989 splits. We do so by ~~evaluating~~ calculating the average TP_{rate} and FP_{rate} over the 989 ROC curves at every discrimination threshold value. ~~By calculating the~~ The average of the 989 values ~~we produce~~ represents a single point on the *threshold-averaged ROC curve*.

We repeat this process after adding each time one more feature to ~~every tuple~~ the tuples in the training dataset~~, in~~. We add the features in the decreasing order of the information gain value. For example, if before we grew a classification tree only with *mean_nrPrefs*, in this step we do so by using the *mean_nrPrefs and* the *mean_MonitorsDetecting*, which is the feature with the second-highest value of information gain. We then repeat the process explained above for deriving the threshold-averaged ROC curve for each subset of features. We depict in Figure 4 the threshold-averaged ROC curve for every of the 9 different subsets of features considered for the tuples in the known dataset used for training.

To identify which is the optimal set of features, we compare the AUC for the 9 different threshold-averaged ROC curves in Figure 4. We observe that the classification tree using the first 7 most-important features has the highest performance, with an average AUC equal to 0.94. In the overall best operating point for all the 989 data splits, the decision tree has an average TP_{rate} equal to 0.99 and an average FP_{rate} equal to 0.1.

2) Boosting for Improved Accuracy: We previously ~~found that~~ find that the optimal decision tree uses only the first 7 most-important visibility features (as per the information gain value included in Table II) to classify the *LVPs*. We now move on to the second phase of the decision tree induction present in Algorithm 1, namely boosting the base tree model.

Boosting is one of the most powerful learning mechanisms proposed in the last 20 years, useful to improve the accuracy of a classification algorithm [22]. The main idea behind this algorithm is to combine many base classifiers (e.g., in our case, the CART decision tree built with 7 features) to produce one robust classification algorithm. Unlike other boosting algorithms, AdaBoost [18] adjusts adaptively to the errors of the base learners derived at each iteration. We use the AdaBoost.M1 algorithm ~~implemented in~~ available from the publicly available package *adabag* [23] for R.

In order to improve the classification performance across all the possible data splits, we combine 50 such base learners using the boosting ensemble technique. We ~~choose to~~ run 50 boosting rounds to make sure that we do not over-fit the algorithm to the training data. After running experiments with variable numbers of boosting iterations, we find that a number of 50 boosting rounds improves the overall classification performance without over-fitting the classification tree.

To guarantee a good general performance of the boosted tree-based model with 7 features, we determine next the overall optimal discrimination threshold for ~~all~~ the 989 splits. For each of the 989 boosted decision trees obtained, we derive the associated ROC curve, to obtain an aggregated view of the performance of each classifier. ~~Since we aim to determine the optimal operating point over *all* the 989 models, we~~ We then calculate the *threshold-averaged ROC curve* ~~for~~ of the 989 ROC curves.

In Figure 5 we depict the resulting averaged ROC curve, which we further use to calibrate the model. We first note that, independently of the threshold value, the classification model is generally very accurate for any of the training-validation splits, with an AUC equal to 0.997. Moreover, we observe that in the best operating point, the decision algorithm has an average true positive rate equal to 0.98, and an average false positive rate of 0.05. The average accuracy of the decision model is 98%. Though this is a very positive result, our aim is to design a classification algorithm which generalizes by accurately predicting for *any* previously unknown case of AS originating *LVPs*. Given that for a new AS we do not have ways to learn the distribution of intended and unintended prefixes, we choose as optimal operating point the value of the threshold where the performance of the algorithm is the highest for any possible distribution of prefixes per class. In other words, we choose the value of the threshold which gives the best performance under the worst known conditions. This point is the break even point, where the threshold value is equal to 0.6. In this operating point, the decision algorithm

has an average true positive rate equal to 0.99, and an average false positive rate of 0.24. The average accuracy of the tree-based model at the break-even point is 95%.

Though we observe a slightly weaker performance here than in the best operating point, we ensure that the decision algorithm at the break even point achieves optimal performance for new cases of ASes originating *LVPs*. We further refer to the boosted tree-based classification model using the 7 most-important features and operating with a discrimination threshold of 0.6 as the *Winnowing Algorithm*.

3) Testing on the Hold-Out Dataset: To further estimate the Winnowing Algorithm performance, we test the model on the hold-out independent dataset, which we did not use in the training/validation phase. First, we train the prediction model on all the available ground-truth data, encompassing both validation and training datasets. We then test the boosted decision tree on the tuples from the hold-out dataset. The performance of the winnowing algorithm is characterized by an average true positive rate of 0.951, with a 95% confidence interval of [0.87, 0.99] and an average false positive rate of 0.01, with 95% confidence intervals of [0, 0.02]. We further calculate the accuracy of the Winnowing Algorithm, by evaluating the overall proportion of tuples correctly classified. The average accuracy on the hold-out test set is 97.2%.

D. Discussion on the Machine Learning Approach

Though the machine learning approach is gaining popularity for Internet-oriented applications, it is sometimes hard to understand the functionality of the mechanism. In this section, we provide the intuition behind the decision rules implemented in the winnowing system.

1) On the Visibility Features: One particularity of the Winnowing Algorithm is that it only uses visibility features of the ground-truth *LVPs* for classification. This set of features is consistent with the operational status of the routing system. For example, accidental routing leaks usually generate a large number of prefixes at once. This explains why, in the context of the Winnowing Algorithm, the most important feature used to pinpoint *unintended LVPs* is the average number of *LVPs* injected by the same origin AS. Also, a high variation in the total number of *LVPs* from the same origin AS hints that the prefixes may not be stable expressions of long-lived routing policies, but side-effects of routing errors. Additionally, we use as features the per-LVP visibility degree and the proportion of active monitors which "see" the *LVPs*. These two features capture the prefix visibility dynamics caused by variations in the daily set of active monitors available. We observe that majority of unintended LVPs have a stable visibility degree of 3, which is consistent with the fact that misconfiguration usually affect the routing policies of the ASes in the ~~direct~~ vicinity of the origin. Furthermore, discarding the last two features, namely the *TimeActive* and the *VisibilityLabel*, is also justified. For instance, as previously depicted in Figure 2, the lifetime of *unintended LVPs* is longer than the lifetime of easily-noticeable anomalous events, which are quickly fixed by the origin. For this reason, the lifetime of unintended LVPs is consistent with the lifetime of intended LVPs which appear as a result of the correct routing policies. Thus, *TimeActive* does not discriminate well between the two classes of *LVPs*.

2) On the Data Structure: One of the restrictions we impose in the data structure proposed in the machine learning study design is ~~that~~ to include all the *LVPs* originated by the same AS ~~be all included~~ in the same dataset, namely training, validation or hold-out test data. This restriction ensures that we correctly design the winnowing mechanism to distinguish between *LVPs* from **new ASes** that might be suffering from unforeseen events. However, it is also important to accurately classify **new *LVPs*** from a network which already provided feedback used for deriving the Winnowing Algorithm. In order to verify the performance of the model for such cases, we perform a very simple experiment. Namely, we split the dataset independently of the origin AS. We withhold 100 random instances of each class for testing and use the rest for training. We find that the Winnowing Algorithm derived in Section IV-C2 performs a highly accurate classification of the test samples, only misclassifying one out of the 200 tuples. In other words, when training on *LVPs* originated from one particular origin AS, the algorithm has a fairly easy task in classifying other new LVPs originated by the same AS.

3) On the Ground-truth Lifetime: The ground-truth dataset of 20,000 *LVPs* documents a wide range of cases of ~~previously~~ long-lived undetected anomalous events ~~, affecting the interdomain~~ that are affecting the Internet entities. Though the causes for the anomalies we detect are recurring in the Internet, their appearance in the BGP Visibility Scanner may change in time. It is unclear at this point how the validity of the ground-truth dataset and, consequently, the performance of the Winnowing algorithm would be impacted by the evolution in time of ~~the~~ routing anomalies and of the routing system itself. Additional amount of feedback from active users of the visibility scanner can advance our understanding of the evolution of LVPs in the Internet. Furthermore, a ground-truth dataset covering a longer period of time would also allow us to enhance the capabilities of the Winnowing Algorithm, since this would offer ~~many different~~ more examples of intended and unintended LVPs, while also capturing the evolution in time of the visibility parameters ~~of~~ for routing anomalies. We leave for future work the analysis of the lifetime of the ground-truth knowledge we ~~have accumulated~~ accumulate and the stability of the accuracy of the winnowing system in time.

V. Reachability and Visibility of Prefixes

~~With~~ Using the Winnowing Algorithm ~~we propose in~~ from the previous section, we are able to accurately distinguish unintended LVPs, which unexpectedly emerge in the Internet because of configuration errors or bogus routing policies ~~tacking~~ taking effect. This set of anomalies not only impacts the efficacy of the intended routing policies of the ASes affected. There are cases when the LVPs become globally unreachable, since there might not always be a less-specific covering HVP to ensure that the destinations attached are globally reachable. In this section, we aim to establish if prefix visibility at the interdomain level can be further used to alert ISPs about reachability issues their prefixes might be suffering.

In other words, we analyze next if the routing anomalies that render a prefixes as LVP can also deteriorate the reachability of the corresponding address space. To this end, we perform active reachability measurements towards prefixes from all of the three visibility classes, i.e. HVP, LVP and DP~~, with the goal of establishing~~. Our goal is to further verify the existence of a correlation between visibility and reachability of prefixes in the Internet. For this analysis, we focus on the set of IPv4 and IPv6 LVPs ~~derived on~~ from the 8th of August, 2013.

A. Measurement Approach

We begin our analysis by presenting the approach we propose to determine if a prefix is reachable from a given vantage point in the Internet. Given the current stage of density of the IPv4 Internet, this should not constitute a concern when testing the reachability of the address space. ~~HowevverHowever~~, the challenge comes from ~~doing~~ testing this for IPv6 prefixes, for which it is not a simple task to find an address that is actually allocated to an active host in any prefix. For consistency, we further design and employ the same measurement approach both for IPv6 and IPv4 prefixes.

The idea we put forward is to probe the reachability of a prefix with traceroute towards a random address ~~within~~ contained by the prefix. We can then check if the last node replying to the traceroute belongs to the AS of the target prefix or to one of its Internet providers, as observed in the BGP AS-Path. In other words, our measurement approach is as follows.

- We send ICMP traceroute probes towards an IP address in the target prefix.
- We say that the target prefix is reachable if :
  1) The traceroute probe reaches the origin AS of the target, i.e., the last AS which appears in the BGP AS-Path[7].
  2) The traceroute probe traverses the second-last AS in the BGP AS-Path[8].

Traceroute is one of the most widely used network measurement tools, useful both to network operators and researchers. Apart from the default traceroute [24] , several other traceroute approaches are available, namely UDP traceroute, TCP traceroute and ICMP traceroute. The traceroute probing method we employ is ***ICMP traceroute***, which has been previously shown to be the most successful approach in terms of replies received [25] .

We include the ~~latter hypothesis~~ second hypothesis in our measurement approach because there may be cases where, even if the probe does reach its destination, it might happen that the AS of the source IP for the last ICMP message received is actually the transit provider of the target AS. This happens because it is a common operational practice that ASes use addresses from their providers for their transit links. As a result, the router within the destination network that issues the last message of the traceroute ~~process~~ will do so using ~~an source address from its ISP's address space~~a source address that belongs to its ISP. This may also be due to reachability problems in the last hop, which our methodology is unable to distinguish.

~~We establish next which of the most popular traceroute approaches is also the most efficient. Traceroute is one of the most widely used network measurement tools, useful both to network operators and researchers. Apart from the default traceroute [24] , several other traceroute approaches are available, namely UDP traceroute, TCP traceroute and ICMP traceroute. The traceroute probing method we employ is the ***ICMP traceroute***, since it has been previously shown to be the most successful approach in terms of replies received [25] .~~

B. Validating the Measurement Methodology

We validate our approach by checking the reachability status of a set of prefixes which are a-priori known to contain at least one reachable address. Our aim is to determine if the reachability of a randomly chosen IP address is representative for the most-specific covering prefix. We use a control set of 70.000 addresses which are known to be reachable. This is made up of addresses from many sources, including DNS entries, Alexa's top sites, and several other sources. For each of these addresses, we retrieve the most-specific covering prefix installed in the BGP routing tables. We use public routing data information to determine the most-specific prefixes covering each of these reachable addresses. The set of prefixes determined represents address space known to contain at least one address that replies to traceroute probing. These prefixes form the control set of prefixes which we use for validation.

We test the methodology from a machine within a major Japanese ISP, for which we also have the corresponding BGP routing data. We send ICMP traceroute probes towards **the first IP address that is different from the address known to be reachable** within each of the prefixes previously inferred. According to the proposed methodology, we consider that the traceroute probe reaches the destination when it traverses either the origin AS of the target, either the second-last AS (2LH) appearing in the BGP AS-Path towards the target. In order to identify the 2LH towards a prefix, we analyze the AS-Path information in the BGP routing table of the AS from which we are generating the traceroute messages.

After parsing the results of our traceroute tests, we learn that the ICMP traceroute probes successfully reached more than 96% of these *a-priori* reachable prefixes. Consequently, the methodology we propose is able to identify with 96% accuracy the reachability status of an IPv6 prefixes. For the other 4% of prefixes, our methodology is unable to determine reachability. This may be due to several reasons, including ICMP filtering or routers silently discarding packets.

C. Reachability of Visibility Classes

We further aim to establish the reachability for prefixes ~~with all of~~ within each the three different ~~classes~~ categories of interdomain visibility, namely DPs, non-dark LVPs and HVPs. We perform the reachability measurements first from

[7]Usually, in the BGP AS-Path the last hop represent the origin AS of the prefix, while the first hop represents the AS whose routing table we analyze.

[8]Following the order of the ASes in the BGP AS-Path attribute, the second-last hop (2LH) in the AS-Path corresponds to the transit provider of the origin AS.

a single source, for which we also know the state of the BGP routing at the moment of testing. From the point of view of the measurement source, the High-Visibility Prefixes are the prefixes contained in its BGP routing table. There are in total 13,195 such IPv6 prefixes and 480,400 IPv4 HVPs. These prefixes may not be globally High-Visibility, since there may be other routing tables not "seeing" some of these prefixes. We label all the rest of prefixes learned from the rest of the routing tables collected from the public repositories as Limited Visibility. ~~The latter~~ LVPs reach a total number of 2,359 prefixes for IPv6 and 87,397 prefixes for IPv4 at the time of the analysis. In order to check if any of the Limited Visibility prefixes are in fact Dark Prefixes from the point of view of the ISP, we check which LVPs have a less-specific HVP in the ISP's routing table to offer global reachability. We are thus able to single out a total number of 511 IPv6 DPs and ~~2917~~ 2,917 IPv4 DPs.

In the case of the IPv6 LVPs which have a covering high-visibility IPv6 prefix (i.e., they are not dark), we observe that 94% of the prefixes are reachable from the Japanese ISP's network. For the IPv4 LVPs with covering HVP, we find that 89% were reachable from the single vantage point used. This is consistent with the precision of our tool~~so~~, thus we cannot make any claims about reachability problems in the LVP set. We next evaluate the reachability status for the IPv6 and IPv4 DPs. We learn that for more than 95% of these IPv6 prefixes, the traceroute ~~measurements~~ probes did not reach the target. Consequently, less than 5% of the dark address space is reachable from the vantage point. This result is further consistent with the reachability measurements performed for the IPv4 dark prefixes, out of which less than 3% were reachable from the vantage point. This results shows that, contrary to the case of non-dark LVPs, where there might be a less-specific covering HVP to ensure global reachability, the DPs do present serious connectivity problems, when measured from this single vantage point.

D. RIPE Atlas Measurement Methodology

Previously, we have seen that, because of the lack of a covering HVP, dark prefixes exhibit serious connectivity issues, when tested from a vantage point in the Internet. In this section, we further test if this is globally valid. We use the RIPE Atlas Platform [26] to run ***large-scale measurements for characterizing the reachability of the dark address space***.

We zoom out from the previous localized analysis of reachability, and test the reachability of the DPs from 100 different probes active in the RIPE Atlas platform. We run the measurements both towards IPv6 and IPv4 dark prefixes. More specifically, we test 473 IPv6 DPs derived from analyzing 110 IPv6 GRTs and 3,200 IPv4 DPs derived from analyzing 154 IPv4 GRTs. We send ICMP probes towards a target IP address within each of the v6 and v4 DPs.

We verify the reachability results in accordance with the methodology proposed in Section V. Point (2) of the proposed methodology requires to verify if the traceroute probe traverses the provider of the origin AS for the target prefix. As opposed to the previous case where we have the BGP routing table from the AS hosting the traceroute source to analyze, we now do not have access to the BGP routing tables corresponding to the 100 Atlas probes used. In order to overcome this issue, we build a set of *probable* second-last hops, which are likely to be traversed towards each of the possible destination ASes. We do so by analyzing all the available routing tables from the ASes active in RIPE RIS and/or Routeviews, and monitoring the ASes appearing as 2LHs towards the origin AS of the target prefix. We then state that the target prefix is reachable if *the traceroute probe traverses* ***any*** *of the probable second-last ASes towards the origin AS of the target prefix.*

In order to further understand the impact of relaxing point (2) of our methodology, we perform the following verification. We first determine the set of probable second-last hops towards every destination AS, without using the BGP routing information from the AS hosting the machine we used to run the traceroute tests in Section V-C. We then verify the proportion of prefixes for which the 2LH appearing in the BGP AS-Path from the Japanese AS routing table is *not* among the set of 2LHs likely to be traversed towards the target prefix. We find that only for 0.05% of the targets, the 2LHs appearing in the BGP AS-Path of the Japanese ISP are not included in the set of probable 2LHs derived from all the available GRTs.

E. Results

After processing all the traceroute measurements from each of the 100 probes towards the Dark Prefixes, we conclude that the average reachability degree[9] for an IPv6 DP is of 46.5%, whereas for an IPv4 DPs it decreases to only 17.4%. To further understand this result, we verify how the DP reachability correlates with the visibility degree of a DP. We show in Figure 6 the scatter-plots both for IPv6 and IPv4 DPs' reachability against their visibility within the corresponding sample of ASes analyzed. We observe that for the IPv6 DPs, depicted in the left-side plot, there is a stronger correlation between reachability and visibility than for the IPv4 DPs. This happens because, for the IPv4 DPs, we see a high number of prefixes with very limited visibility, but which are highly reachable from the sample of 100 probes chosen. We observe that in the v4 plot from Figure 6 there are approximatively 8% of IPv4 prefixes with visibilities smaller than 0.2 and reachability larger than 0.2. As previously noted in [27], this may be due to default routing in IPv4 or static routing [28]. In Section III, we explain many of the real-life operational reasons for which this type if IPv4 DPs emerge in the Internet. For example, we observe in the lower-left corner of the IPv4 plot in Figure 6 a very large number of ~~v4DP~~ IPv4 DP (approximatively 72% of all the IPv4 DPs) with a reduced visibility degree and a corresponding low reachability degree. According to the observations in Section IV-D regarding the intuition behind the visibility features used by the Winnowing Algorithm, a small visibility degree of the LVPs indicates the fact that the IPv4 DPs we test may be effects of misconfiguration or bogus routing policies. For example, these IPv4 DPs may be route leaks which, as we learn from the operational use

[9]We define the reachability degree as the number of probes out of the 100 Atlas vantage points that successfully reach the traceroute target.

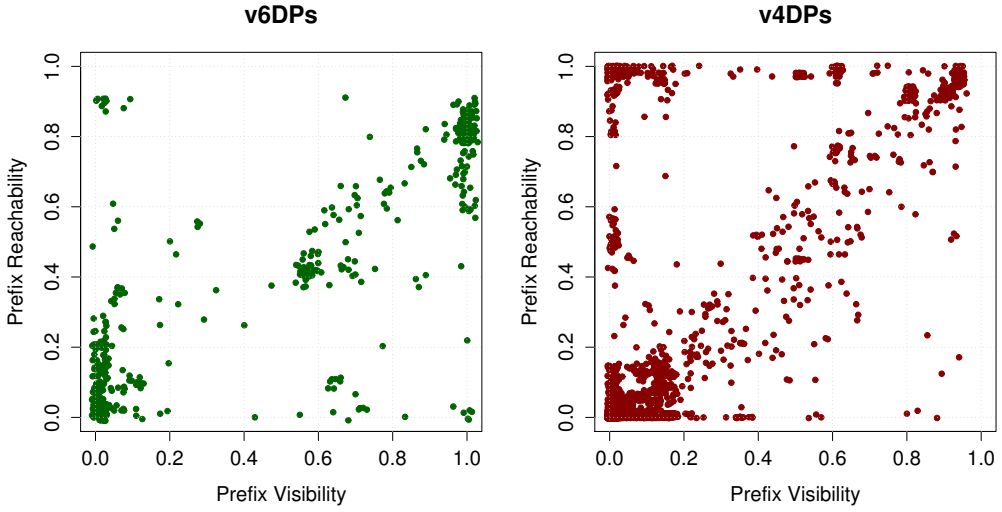

Fig. 6. Scatterplot of reachability probability against the DP's visibility, for IPv6 DPs and for IPv4 DPs.

cases explained in Section III, often occur in the Internet. Consequently, the lack of reachability observed for IPv4 DPs is largely explained by the fact that these prefixes are unintended to be visible in the Internet to begin with. At the same time, even if the IPv6 DPs do not follow the known symptoms of route leaks or anomalies previously learned from the IPv4 cases, they do struggle with important lack of reachability. This further shows that, while in IPv4 the DPs are in majority results of mistakes or slips in the network configuration, in IPv6 the DPs might appear as side-effect of the early stages of development of the network.

Next, we focus on analyzing the reachability of DPs whose ASes originate both IPv4 and IPv6 dark address space. To this end, we separate only the DPs which are generated by ASes active both in v4 and v6. In total, we test the reachability of 214 dark prefixes, out of which 88 are IPv6 DPs. We learn that, in average, for the IPv6 DPs there is an average probability of 40% of being reachable from a vantage point in the Internet, which is consistent with the general reachability result for all the IPv6 DPs. This probability decreases to 20% for the IPv4 DPs, also consistent with the overall reachability result for IPv4. Consequently, there is no apparent correlation between the reachability of IPv4 DPs and IPv6 DPs originated from the same AS, these actually following the general reachability trends previously established.

VI. Related Work

Most of the work related to our efforts tackle the analysis of BGP raw data, which can be tricky and difficult. There are numerous efforts towards detecting security related routing conditions, such as prefix hijacking (e.g., PHAS [29]). Also, various tools exist to provide useful information for operators [30], [31]. Multiple operational misconfiguration have been reported [4], but attempts go far beyond this. They include *RIPE Labs* [9], which has a whole section devoted to tools that assists operators or Renesys [32] and BGPmon [33], which operate this type of services to operators for a fee. Unlike tools which integrate a vast amount of operational problems [8], we do not focus on inferring and/or monitoring the AS-level topology of the Internet, but on monitoring the healthy deployment of routing policies through prefix visibility. In this sense, our work is very closely related to the work on BGP wedgies by Griffin et al [6], [34]. However, none of those theoretical work is able to detect problematic routing conditions based on raw BGP observations. All of this work requires access to configuration files, which are typically not shared. The latter are considered a company secret which BGP was designed to hide, making it hard to be inferred [35]. While we understand the limitations of BGP protocol monitoring, we noticed that still a great deal that can be inferred. In this sense, our work aims at reporting and aggregating the information to make it usable for operators. Despite that many other similar tools [8], [9] leverage the massive amount of available routing data, the BGP visibility scanner is, to the best of our knowledge, the only tool offering specific information on global prefix visibility.

Machine learning in the context of interdomain routing has been already proven to be a successful approach. Using traffic feature distributions, Lakhina et al. [36] show that the existence of some anomalies can be detected from traffic flows. Furthermore, a Bayesian framework has been previously proposed for detecting mistakes in the router configuration files using statistical anomalies [37]. Relying on network data, the usage of statistical algorithms has been advanced to detect deviations from the long-term profile of BGP routing updates [38]. Similarly, an instance-learning framework has been previously recommended for identifying deviations from the normal defined state of BGP routing dynamics [39]. Likewise, Li et. al advance a rule-based framework for the detection of abnormal routing behavior caused by a major worm or a blackout [40].

During the past years, IPv6 has received a lot of attention both from the operational and the research community. Various related works look into the transition of the IPv4 network infrastructure to IPv6 [41] and how the Internet topology, routing and performance across the two compares [42]. It had been proven that the routing dynamics in the IPv6 topology are largely similar to those know from IPv4, even if the degree of IPv6 deployment is still far behind the IPv4 expansion [42].

VII. Conclusions

The BGP Visibility Scanner has proven its ability to trigger valid visibility alarms and to help operators debug their routing policies. We were able to help identify more than *18,000 unintended LVPs* and assist the origin networks in identifying their causes. Such prefixes can be easily missed as they are often overlooked as valid expressions of intentional events. For example, an ISP was able to learn that 4,000 of its prefixes were leaking through some of its direct peers and were visible in the Internet since at least 6 months before the query was performed. Such events may stem as a consequence of the merger between large ISPs whose configurations are consequently changing. This type of transition may affect the visibility of some prefixes, as it has been observed in the case of the Level3-Global Crossing merger [43].

In light of the observed perpetuity of such anomalous interdomain events, we learn that there is an overall acute need for a simple warning system for faulty configurations and/or problematic external routing conditions to assist operators in optimizing the performance of their routing policies. We thus rely on machine-learning to design a *Winnowing Algorithm* able to predict with 95% accuracy if a LVP is intended or unintended. We leverage the robust machine learning concept of *boosted classification trees* [18] to train the system on ground-truth *LVPs*, and thus enable it to learn the patterns of misconfiguration and bogus routing policies which are normally hard to detect. Furthermore, the classification model uses only visibility-related per-prefix features in order to predict the class of the *LVPs*.

While affecting the global visibility of prefixes, ~~mis-configured~~ misconfigured or unintended routing policies also impact the global reachability of prefixes. Consequently, we research how the visibility degree of a prefix impacts its global reachability. From multiple vantage points in the Internet, including 100 RIPE Atlas active probes, we test the reachability of both IPv4 and IPv6 LVPs. We find that *limited visibility does not necessarily imply limited reachability*, since there could be a less-specific ~~*HV*~~*HVP* covering the LVP to provide global reachability. However, Dark Prefixes (DPs), which by definition do not have a covering less-specific prefix, remain highly unreachable. Moreover, while the IPv4 dark address space can be largely explained as route leaks or mistakes, this is not valid for the IPv6 DPs. We believe that this is a serious problem for the Internet, as limited reachability of a non-negligible set of prefixes cripples the fundamental function of the Internet, i.e., ensuring global connectivity for every host attached to it.

VIII. Acknowledgements

This work was partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) grant no. 317647 (Leone).

References

[1] "How the Internet in Australia went down under." [Online]. Available: http://www.bgpmon.net/how-the-internet-in-australia-went-down-under/

[2] "Pakistan hijacks YouTube." [Online]. Available: http://www.renesys.com/2008/02/pakistan-hijacks-youtube-1/

[3] R. Bush, "Oh K can you see," in *NANOG mailing list archives*, October 2005.

[4] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, 2002.

[5] L. Quan, J. Heidemann, and Y. Pradkin, "Trinocular: Understanding Internet Reliability Through Adaptive Probing," in *Proceedings of the ACM SIGCOMM Conference*, Hong Kong, China, August 2013.

[6] T. Griffin and G. Huston, "BGP Wedgies," 2005, RFC 4264.

[7] H. Yan, B. Say, B. Sheridan, D. Oko, C. Papadopoulos, D. Pei, and D. Massey, "IP Reachability differences: Myths and realities," in *IEEE Global Internet Symposium (GI 2011) at INFOCOM 2011*, april 2011, pp. 834 –839.

[8] Y.-J. Chi, R. Oliveira, and L. Zhang, "Cyclops: the AS-level connectivity observatory," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 5, 2008.

[9] "RIPE Labs." [Online]. Available: https://labs.ripe.net/

[10] C. Labovitz, A. Ahuja, and M. Bailey, "Shining Light on Dark Address Space," Arbor Netwoks, Ann Arbor, Michigan, USA, Tech. Rep. TR-2001-01, November 2001.

[11] A. Lutu, M. Bagnulo, and O. Maennel, "The BGP Visibility Scanner," in *IEEE Global Internet Symposium (GI 2013)*, April 2013.

[12] "BGPMON Alert Questions." [Online]. Available: http://mailman.nanog.org/pipermail/nanog/2014-April/066171.html

[13] A. Lutu, M. Bagnulo, J. Cid-Sueiro, and O. Maennel, "Separating wheat from chaff: Winnowing unintended prefixes using machine learning," in *Proceedings of 33rd IEEE International Conference on Computer Communications*, ser. IEEE INFOCOM 2014, April 2014.

[14] A. Lutu, M. Bagnulo, C. Pelsser, and O. Maennel, "Understanding the reachability of ipv6 limited visibility prefixes," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, 2014, vol. 8362.

[15] "BGP Routing Table Analysis Report." [Online]. Available: http://bgp.potaroo.net/

[16] "Team Cymru - The Bogon Reference." [Online]. Available: http://www.cymru.com/BGP/bogons.html

[17] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "An analysis of BGP multiple origin AS (MOAS) conflicts," in *1st ACM SIGCOMM Workshop on Internet Measurement*, 2001.

[18] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, ser. EuroCOLT '95, 1995.

[19] J. Han, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[20] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[21] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/

[22] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer-Verlag, 2001.

[23] E. Alfaro-Cortes, M. Gamez-Martinez, and N. Garcia-Rubio, *adabag: Applies multiclass AdaBoost.M1, AdaBoost-SAMME and Bagging*, 2012.

[24] V. Jacobson, "traceroute." [Online]. Available: ftp://ftp.ee.lbl.gov/traceroute.tar.gz

[25] M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute probe method and forward ip path inference," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, 2008.

[26] "Ripe Atlas." [Online]. Available: https://atlas.ripe.net/

[27] R. Bush, O. Maennel, M. Roughan, and S. Uhlig, "Internet optometry: Assessing the broken glasses in internet reachability," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '09, 2009.

[28] Y. Shavitt and N. Zilberman, "Arabian nights: measuring the arab internet during the 2011 events," *Network, IEEE*, vol. 26, no. 6, pp. 75–80, November 2012.

[29] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: a prefix hijack alert system," in *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, 2006.

[30] J. Wu, Z. M. Mao, J. Rexford, and J. Wang, "Finding a needle in a haystack: pinpointing significant BGP routing changes in an IP network," in *Symposium on Networked Systems Design & Implementation*, 2005.

[31] "BGP Routing Leak Detection SystemRouting Leak Detection System." [Online]. Available: http://puck.nether.net/bgp/leakinfo.cgi

[32] RENESYS, http://renesys.com/.
[33] "BGPmon." [Online]. Available: http://www.bgpmon.net/
[34] D. Perouli, T. Griffin, O. Maennel, S. Fahmy, C. Pelsser, A. Gurney, and I. Phillips, "Detecting Unsafe BGP Policies in a Flexible World," in *International Conference on Network Protocols (ICNP)*, 2012.
[35] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush, "10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems," *Selected Areas in Communications, IEEE Journal on]*, vol. 29, no. 9, 2011.
[36] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proceedings of SIGCOMM '05*, 2005.
[37] K. El-Arini and K. Killourhy, "Bayesian Detection of Router Configuration Anomalies," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, ser. MineNet '05, 2005.
[38] K. Zhang, A. Yen, X. Zhao, D. Massey, S. F. Wu, and L. Zhang, "On Detection of Anomalous Routing Dynamics in BGP," in *NETWORKING*, 2004.
[39] J. Zhang, J. Rexford, and J. Feigenbaum, "Learning-based anomaly detection in BGP updates," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, ser. MineNet '05, 2005, pp. 219–220.
[40] J. Li, D. Dou, Z. Wu, S. Kim, and V. Agarwal, "An internet routing forensics framework for discovering rules of abnormal BGP events," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, October 2005.
[41] M. Nikkhah, R. Guérin, Y. Lee, and R. Woundy, "Assessing ipv6 through web access a measurement study and its findings," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, ser. CoNEXT '11, 2011.
[42] A. Dhamdhere, M. Luckie, B. Huffaker, k. claffy, A. Elmokashfi, and E. Aben, "Measuring the deployment of ipv6: topology, routing and performance," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, ser. IMC '12, 2012.
[43] "3356 leaking routes out 3549 lately," in *NANOG mailing list archives*, March 2014.