BGP Route Leaks Detection at Scale

Julie Rossi
ANSSI
Paris, France
julie.rossi@ssi.gouv.fr

Guillaume Valadon
ANSSI
Paris, France
guillaume.valadon@ssi.gouv.fr

ABSTRACT

The BGP routing protocol is vital to the Internet and does not include any security mechanism. Any AS (Autonomous System) can advertise whatever prefix it wants, no matter if it actually belongs to another. This allows malicious behaviors and does not prevent configuration errors. To help distinguish them, we implemented a method to detect route leaks, a typical operational mistake. A route leak occurs when an AS advertises, usually by accident, some prefixes it learned from others. As the leak gets propagated, it can have a major impact since traffic will be directed toward the AS having made the leak instead of the legitimate AS. This phenomenon was noticed several times along the past years, sometimes impacting thousands of ASes. In order to identify route leaks, we defined a new methodology, and applied it on 2014, 2015 and 2016 BGP data. It successfully detected 6, 36 and 22 route leaks, respectively. This represents tens of thousands of leaked prefixes each year. In 2016, we detected that there are more than 11 000 conflicts due to route leaks. For better understanding and reproducibility, we published both our implementation code and the input datasets at <https://github.com/ANSSI-FR/route_leaks>.

CCS CONCEPTS

• **Networks → Routing protocols**;

KEYWORDS

BGP, leaks, detection

ACM Reference format:
Julie Rossi and Guillaume Valadon. 2017. BGP Route Leaks Detection at Scale. In *Proceedings of The 13th International Conference on emerging Networking EXperiments and Technologies, Seoul, South Korea, December 2017 (CoNEXT 2017),* 7 pages.
DOI: 10.475/123_4

CoNEXT 2017, Seoul, South Korea

DOI: 10.475/123_4

1 INTRODUCTION

Each Internet operator manages sets of contiguous IP addresses called prefixes. To form the infrastructure of the Internet, the operators connect to each other using the BGP protocol [17]. The aim of this protocol is to exchange reachability information about the prefixes between two operators, called Autonomous Systems (ASes) and identified by a unique number. Each AS advertises to its peer that it can route traffic to its own prefixes, as well as those of its customers.

BGP does not include any authentication method for prefix advertisements. Consequently, any AS can advertise whatever prefix it wants, including prefixes that actually belong to another. Thus, a prefix conflict occurs when two ASes announce overlapping prefixes at the same time. Those conflicts can be either innocuous or malicious, the latter being considered as a prefix hijack.

In fact, both hijacks and route leaks are conflicts. However, unlike prefix hijacks, route leaks appear after BGP router configuration errors, when an AS originates a lot of prefixes belonging to other ASes. Leaks are characterized by many conflicts originating from the same AS over a short period, and affecting many ASes at the same time. Detecting BGP route leaks at scale is a challenging task, because it involves monitoring a lot of BGP messages from various locations.

In this paper, we propose a new algorithm to automatically detect route leaks while minimizing false positives. Based on our observations, we modeled a leak as an event in which a significant increase in the number of prefixes advertised by an AS and the number of ASes in conflict with this AS simultaneously, occur over a short period. The algorithm builds on our previous empirical methodology [1] to find out the most efficient parameters by comparing their detection performances. We provide detection results over 2014, 2015 and 2016. They are significant due to the large datasets involved: each year accounts for 450GB of raw BGP messages. As such, this paper makes three distinct contributions: (1) description and comparison of a new methodology (2) publication of the datasets used to perform the detection, and (3) release of the tools implementing the methodologies. The latter allows the community to reproduce and challenge our results.

The generic problem of route leak detection is quite similar to peak detection in a series [10, 15, 19]. However, solely detecting peaks in an abstract series induces a lot of false positives. For example, an operator could deaggregate prefixes

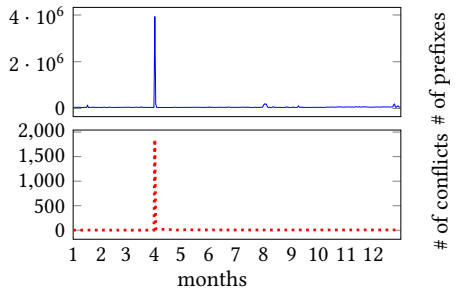


Figure 1: Route Leak Example - AS4761, April 2, 2014

resulting in a high number of prefix advertisements that will likely be detected as a peak. In order to cope with the specificities of the BGP ecosystem, a dedicated algorithm must be used. So far, several [11, 12] were defined by the operators and the academic communities. They use operational knowledge to apply successive filters to series in order to enhance the detection. Our approach is closer to a domain-specific peak detection crafted to detect route leaks efficiently.

This article is organized as follows. First, we present several well-known public examples. Then, we discuss related work, our route leak definition and methodologies. Finally, we describe the resulting algorithm, and the results.

2 ROUTE LEAK EXAMPLES

According to RFC7908 [20], a route leak is the propagation of BGP routing announcements beyond their intended scope. Based on the analysis of publicly documented events, six different types of route leaks were defined. This article only focuses on *Type 5*, named *Prefix Re-origination with Data Path to Legitimate Origin*. It happens when an AS learns some prefixes from its peers and announces them as if they were being originated by it, usually because of routers misconfiguration. From an external observer, a route leak has two noticeable effects since the leaker AS: (1) increases the number of prefixes it announces, and (2) enters in conflict with the legitimate ASes that also announce those prefixes.

Figure 1 represents the behavior of Indosat (AS4761) during 2014 using two time series. On April 2, it originated a route leak [9] that greatly impacted the Internet. Those time series are: (top) the number of prefixes announced by AS4761, and (bottom) the number of ASes in conflict with AS4761. In the figure, obvious peaks appear simultaneously for both prefix and conflict series at the beginning of April. During the leak, the number of announced prefixes reaches 3 000 000, while the conflict count is close to 2000. Usually, the corresponding mean values are 50 000 and 6.

RFC7908 gives some concrete examples of recent route leaks. For instance, China Telecom (AS23724) announced over 50 000 different blocks of IP addresses on April 8, 2010 [8]. This incident impacted tens of thousands of networks over 170 countries. Similarly, Dodo (AS38285) [5] advertised 400 000 prefixes on February 2012. More recently, Innofield (AS200759) and Hathway (AS17488) leaked 320 000 and 400 000 prefixes respectively, in April 2016 and November 2015. These route leaks share the same characteristics: a high number of prefixes are incorrectly originated during a short period.

Throughout the article, plots similar to Figure 1 will be used to represent the ASes behaviors. They are scaled to ease the peaks visualization.

3 RELATED WORK

While the route leak phenomenon was officially defined in June 2016 in RFC7908, it has been known for years. Hence, some detection or prevention methods exist both in the literature, and as commercial services or implementations.

Several online monitoring systems [6, 16, 21] detect prefix conflicts in BGP, from one AS point of view. Each time a conflict is detected, the prefix owner receives an alert, and can take corrective measures, if need be. This is a start for route leak detection, but an AS cannot know if the conflict is due to a route leak or another reason, and only the AS impacted by the route leak will be alerted.

In 2010, Ju et al. [12] described an algorithm to detect route leaks. It relies on successive filters using several datasets external to the BGP routing plane, which are usually incomplete. First, it detects all changes of AS origins in BGP advertisements. Then, the algorithm discards these changes based on heuristics such as the advertisement lifetime, prefixes overlapping, BGP session stability, the relation between ASes (using the `whois` contact email), and a list of IXP prefixes. Finally, when an AS is in conflict with more than 10 other ASes, the change is considered to be a route leak. This detection algorithm was applied to BGP data from 2003 to 2009, and detected five to twenty large route leaks every year. Conversely, we propose to only use information from BGP to perform the route leak detection.

Route leaks that we aim to detect imply that the leaker AS will be the route origin. Consequently, leaks are invalid routes, and recent BGP security enhancements can be used to detect them and prevent their propagation. A generalized BGP Prefix Origin Validation, as specified in RFC6811 [14], prevents route leaks by validating routes using RPKI [13]. Unfortunately, as of today, the RPKI repositories are far from being exhaustive and will result in invalidating many routes. For example, in 2015 [1] and 2016, only 10 % of the active French ASes would see all of their prefixes propagated over the Internet, in case of strict RPKI based filtering. On the other hand, BGP configuration best practices [2] strongly advise to use the `maximum-prefix` option to protect BGP sessions. It limits the number of prefixes accepted from a

peer, and therefore represents a weak route leak prevention mechanism.

A naive approach to detect route leaks consists in applying generic peak detection algorithms. We applied several of them [10, 15, 19] to series similar to Figure 1, but none of them successfully detected well-known route leaks. This was expected as their purpose is to detect all peaks present in the series instead of judging whether there is one peak or not.

4 FROM BGP ARCHIVES TO TIME SERIES

We use BGP archives provided by the RIS service from RIPE [18]. Every five minutes, around 20 BGP collectors dump all the BGP messages that they received in the MRT file format [7]. We parse these archives and convert them into the JSON file format using our BGP/MRT parser called MaBo [4].

The next step consists in detecting prefix conflicts. To do so, we developed TaBi [3], a Python module that processes BGP messages and behaves, as close as possible, like a real BGP router. It follows BGP UPDATE and WITHDRAW messages and knows the list of prefixes announced by the 50k ASes at any given time. Using this list, TaBi is able to compare UPDATE messages with the current RIB to detect conflicts.

TaBi processes MRT data on a daily basis, and outputs two files per day: a list of announced prefixes, and a list of conflicts. These files are used to produce three time series per AS: one containing the number of prefixes announced daily, one containing the number of prefixes in conflict, and another one containing the number of ASes in conflict.

5 ROUTE LEAK DEFINITION

We propose to define a route leak originated by an AS as: *a BGP event during which occur **simultaneously** (1) a **significant raise** of the number of prefixes announced and (2) a **significant raise** of the number of ASes in conflict, during a **short period***. Prior to discussing the algorithm design and the choice of its parameters, we will now precisely define the terms short period, significant, and simultaneity.

Short period Based on our observations and strengthened by other studies [12] where all route leaks last less than 25 hours, we assume a leak does not last more than a day.

Significant When representing the number of prefixes or the number of conflicts over the time, we should be able to perceive one abnormal value, much bigger than the others. For example, in Figure 2a, the peak magnitude is limited, hence too small for a route leak and in Figure 2d, the values are big enough, however there are plenty of similar peaks making route leaks unlikely.

Simultaneity The prefixes and the conflicts are analyzed separately, then compared with each other. If a short and significant peak appears in both data on the same day for an AS, as presented in Figure 1, it is considered as a route leak.

Name	Description	Discards
Peak minimum value	Minimal variation at a data point	Peaks too small
Nb of bigger values	Number of points with bigger values	Many big values
Similarity to max	Ratio between the maximum value and the peak	Meaningless peaks
Standard Deviation	Ratio between std with and without the peak	Many variations

Table 1: Parameters Description Summary

6 THE ALGORITHM

In this section, a detailed overview of the route leak detection is first provided. Then, its different parameters are discussed.

6.1 General Principle

The algorithm seeks to determine whether an AS has been the origin of a route leak, during the considered period. To do so, it takes two time series, described in Section 4, as input. The first one is the number of prefixes announced per day by the AS over the year and the second one is the number of conflicts generated per day by this AS over the year. For the conflict series, we chose to use the number of ASes in conflicts, instead of the number of prefixes in conflict. In practice, using the number of prefixes in conflicts does not yield better results than using the number of ASes in conflicts, but it makes the results harder to analyze.

The detection happens in four steps. The first three are run twice, on the prefix and the conflict series. First, all local maximums in the series are selected. Then, each one is tested against the following three criteria. It must (1) have a significant variation from its previous and next values (2) be one of the biggest values of the series and (3) be close enough to the series maximum value. Only those matching all of them are retained. Next, the partial standard deviation computed without the selected maximums should be smaller than the global standard deviation. Finally, peaks found for prefix and conflict data are confronted and those appearing simultaneously in both series are selected as route leaks.

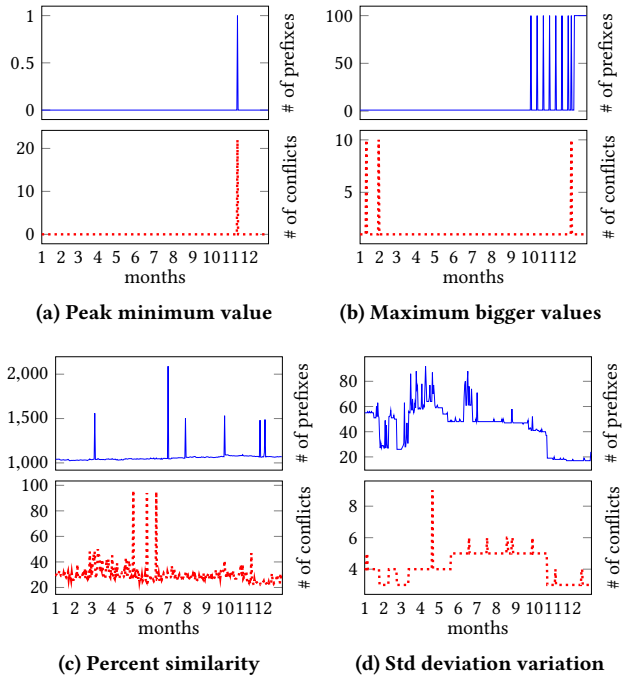
Some of those criteria, such as *significant* or *close enough*, are subjective. They are represented by thresholds that correspond to the algorithm parameters, described below.

6.2 Parameters

The algorithm relies on four[1] parameters. Each represents the threshold used in one step of the algorithm. For convenience, Table 1 summarizes their description.

Peak minimum value The peak should be bigger than its previous and next points with a difference higher than

[1]Actually five, *peak min value* is duplicated between prefixes and conflicts.



(a) Peak minimum value **(b) Maximum bigger values**

(c) Percent similarity **(d) Std deviation variation**

Figure 2: Basic Cases Rejected for Each Parameter

this threshold. This parameter removes series whose peaks are really low, as in Figure 2a. A single prefix cannot be a route leak. This parameter represents data value, thus, it is differentiated between the prefix and conflict series.

Maximum number of bigger values There should not be too many points with a value bigger than the peak. For example, the case in Figure 2b does not look like a route leak but could be detected as such without this parameter.

Minimum similarity to series maximum value The peak value for a route leak should not be too much smaller than the series maximum value. If it is, it no longer seems to be a significant abnormal increase, like in Figure 2c, detected as a route leak if the parameter is not used.

Impact on standard deviation is the maximal ratio between standard deviation computed without the value of the peak being tested and the complete standard deviation. A peak is not retained if it does not impact standard deviation. This would mean that big variations are usual, not anomalies. The plot in Figure 2d is an example of some data fitting every other requirements but still not being an obvious route leak.

7 PARAMETER TUNING

This section describes our goal of avoiding false positives and the methodology used to tune the parameters presented in Section 6.2.

7.1 Goals

The choice of the parameters values is made with the purpose of achieving no false positives while minimizing false negatives and so increase the detection. In our context, a false negative is route leak that is not detected. Conversely, a false positive is an unacceptable error that considers that a meaningless peak is a route leak.

Each parameter represents a reason why a peak should not be detected as a route leak. Their combination, when carefully selected, prevents detecting false positives.

While designing the algorithm, and manually experimenting with parameter thresholds for 2015, we managed to keep the number of false positives low. However, while this empirical method proved that the algorithm works fine on the 2015 dataset, this was not a satisfactory approach. The same thresholds do not act as expected for 2014 and 2016.

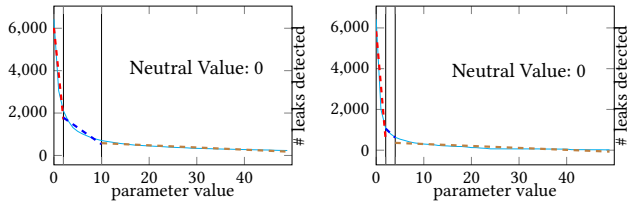
As a consequence, we decided to devise an automatic method to determine the best yearly thresholds. We run the detection algorithm for a whole range of different values for each parameter and study the evolution of the number of cases detected as a route leak in function of the parameter values. The next section describes how this is performed.

7.2 Methodology

We use the following methodology to automatically calculate the thresholds. First, a *neutral value* is defined for each parameter. This is the value that will not impact the detection since it will not discard any case. If all parameters are set to their neutral values, the algorithm will detect every data having simultaneous maximums as as a route leak. The neutral value for each parameter is highlighted in Figure 3.

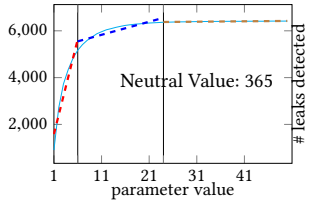
It represents the influence of parameter values on the number of detected route leaks. For each parameter, it displays the number of route leaks detected against its value, while the others are set to their neutral values so they do not impact the result. The 2015 dataset was used since it was the latest available at the time we started the implementation. Yet, we applied the same methodologies to the other two datasets and will discuss the results in Section 8.1. It is important to note that at most 5944 route leaks can be detected in the 2015 dataset. They correspond to simultaneous maximums in the prefix and conflict series.

We used consecutive linear regressions, moving the breaking point along the curve, to model the curve and find out relevant breaking points that significantly change its behavior. They are represented as dashed lines in Figure 3. The question was to determine how many of them were needed to best represent the curves. To answer this, we first tried with two (i.e. one breaking points). For each breaking point tested, we computed the score of both linear regressions and averaged them. Then we picked the best average score. If

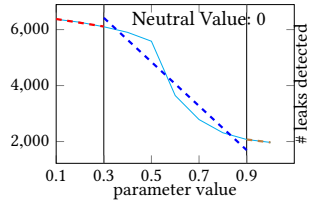


(a) Prefixes peak minimum

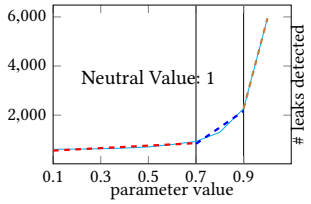
(b) Conflicts peak minimum



(c) Maximum bigger values



(d) Percent similarity



(e) Std deviation variation

Figure 3: Parameter Values with Linear Regressions

it was too low, we restarted the process, using three linear regressions (i.e. two breaking points).

Most of the time, three linear regressions are needed. This offers two possible values for each parameter. We chose to systematically use three linear regressions since offers the choice between limiting the number of false positives and increasing the detection if ones do not mind false positives. We selected the breaking points furthest from the neutral values, the selective ones, except for the *impact on standard deviation* for which we picked the other one. This parameter has a bigger impact than the others so the closest to the neutral value is already quite selective.

8 RESULTS

We ran the algorithm on the 2014, 2015 and 2016 datasets presented in Section 4. In order to assess the algorithm efficiency, we manually classified the 2015 dataset to obtain a ground truth. The corresponding series were categorized either as *route leak*, or *not route leak*.

8.1 Selected Parameter Thresholds

Table 2 displays the values selected for each parameter. They were automatically determined using the methodology described in Section 7. Each run lasts around 45 seconds. The parameter values change for each year, because datasets have different distributions. The goal being to detect anomalies, the most rigorous way to choose the parameters is therefore to fit them to the dataset studied. In practice, trying to detect route leaks in 2015 dataset using 2014 parameters leads to low quality results.

Parameter	Neutral	2014	2015	2016
Peak value - Prefixes	0	24	10	8
Peak value - Conflicts	0	9	4	13
Number of bigger values	365	6	6	8
Similarity to maximum value	0	0.9	0.9	0.9
Standard Deviation ratio	1+	0.8	0.9	0.9

Table 2: Parameter Values

	2014	2015	2016
Number of leaks detected	6	36	22
Potential false positive	0	1	0

Table 3: Detection Results

8.2 Detection Results

The results are summarized in Table 3. With the parameters in Table 2, the algorithm detects 6, 36 and 22 route leaks for 2014, 2015 and 2016, in one minute per year. It corresponds to more than 4 000 000 of prefixes leaked, creating conflicts with almost 25 000 ASes in total. A single AS can impact up to hundreds of other ASes.

A quick visualization of the route leaks detected time series allowed us to validate our results. The goal of zero false positive is almost achieved, only one case on 2015 is potentially one.

The Indosat route leak, described in Section 2 is indeed detected, as well as the other publicly discussed route leaks, like Innofield in 2016 or Bharti in 2015. The algorithm also detected route leaks that did not raise public discussion: on November 16, 2015, the Indonesian AS133354 announced more than 2000 prefixes while usually announcing only 3, entering in conflict with over 600 ASes.

In order to confirm our results, we set up a graphical interface to efficiently annotate the 2015 dataset. First, we automatically removed series that do not have simultaneous peaks, as there should be at least simultaneous local maximums to consider the sample, and those with values fewer than five. Then, we manually classified around 550 cases. It takes less than 5 minutes using the interface to discover 44 route leaks in the dataset. This is an important result that shows that the algorithm successfully classified 80 % of route leaks. Indeed, we annotated fewer than ten events as probable route leaks that the algorithm did not detect. All of them look like real route leaks to an experienced eye, yet exhibits complex patterns such as a duration longer than a day, or

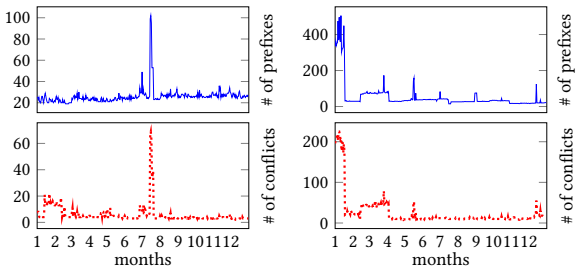


Figure 4: False Negative Examples

non-simultaneous peaks. By definition, the algorithm cannot detect them.

Diving into the results, we observed that route leak real durations vary from less than 1 hour to 24 hours. As two thirds of them last between 15 and 20 hours, it seems that our initial assumption regarding route leak duration, which is less than a day, does not need to be narrowed.

Similarly, the peak values found in the data are continuously distributed, from one to hundreds of thousands preventing us from picking one obvious threshold valid for every situation. This induces that there is no single value that enables route leak detection.

Furthermore, focusing on the ASes detected shows that over the three years studied, ASes have mostly originated one only route leak. This leads to the idea that route leaks are not the fact of a few ASes but can happen to anyone, and reinforce the belief that they are not malicious but are the consequence of configuration mistakes.

8.3 Discussions

To challenge our choice of using the number of ASes in conflicts, the algorithm was instead run on the announced prefix series and the number of prefixes in conflict one, using 2016 dataset[2]. The number of route leaks detected jumped from 22 to 238. All of the 22 ASes were detected by the vanilla algorithm. However the rest is not likely to be route leaks. For example, more than 85 % of them actually impact fewer than five ASes. This confirmed our choice of using the number of ASes in conflict to achieve an efficient detection.

In order to compare our results to the ones of Ju et al. [12] method, we reimplemented their algorithm and ran it on the 2016 dataset. We detect 71 cases as route leaks, almost 75% of them are false positive. This may be due to some misunderstanding or just because the behaviors have changed in the past ten years. Taking advantage of our previous work [22], we decided to modify the described methodology by adding three successive filters. We filtered (1) ASes appearing in the same AS_PATH as the prefix owner, (2) ASes having a route object or ROA corresponding to the announced prefix and (3) when both ASes belong to the same organization (based on whois information). Unfortunately, this modification did not improve the results. Actually, this was predictable as this algorithm does not really take into account the usual behavior of the ASes.

So as to confront our results to real operational issues, we contacted several engineers from hosting services, CDN and network operators, and asked them to comment on the 2016 results. While they all agreed that the 22 do look like route leaks, none of them reported corresponding operational issues. This is a very valuable feedback for our work that motivates us to improve the detection further by looking at the actual propagation of route leaks, as well as publicly reporting our results.

[2]The latest available at this time.

9 CONCLUSION

In this paper, a new solution to efficiently detect route leaks is described. The underlying algorithm was designed to be lightweight and fast in order to process several years of data in few minutes. The results showed that it detected 80% of the 2015 leaks without any false positive. Unlike other similar solutions, our algorithm only relies on BGP messages: no private, or external information are needed.

However, we identified some issues that limit the scope of our solution. When route leaks last for longer than a day or their duration overlap two days the algorithm no not detect them. Moreover, it is not yet ready to be used as an alerting system because it cannot process data in real time.

In our future work, we will focus on improving the detection algorithm and enhancing route leak characterization. For example, we want to study how they are propagated along the Internet by studying related AS_PATH. This will likely help understanding why some leaks do not necessarily cause dramatic operational issues. Moreover, we want to adapt the parameter tuning using a sliding window instead of a fixed yearly duration.

Due to space constraints, we did not discuss our recent work that uses Machine Learning to identify false negatives. The key concept is to run the described algorithm, then use a Machine Learning model to find the missing cases. The current results are not completely reliable but can effectively identify route leak candidates that can be validated manually. It is important to note that using Machine Learning comes at basically no extra cost, as the current algorithm can be used to gather initial labels for the model.

The algorithm, its documentation and the input datasets are available at <https://github.com/ANSSI-FR/route_leaks>. This repository also includes the algorithm by Ju et al. that we used to compare our detection results. The preliminary Machine Learning based algorithm is also included.

REFERENCES

[1] ANSSI. 2015. The French Internet Resilience Observatory. (2015). <https://www.ssi.gouv.fr/uploads/2015/06/internet-resilience-in-france-report_2015_anssi.pdf>.

[2] ANSSI. 2016. *BGP configuration best practices*. Technical Report. ANSSI.

[3] ANSSI. 2016. TaBi. (2016). <https://github.com/ANSSI-FR/tabi>.

[4] ANSSI. 2017. MaBo. (2017). <https://github.com/ANSSI-FR/mabo>.

[5] APNIC. 2012. Leaking Routes. (mar 2012). https://labs.apnic.net/?p=139.

[6] BGPmon. 2017. Route Monitoring. http://bgpmon.net/services/route-monitoring/. (2017). Visited June 2017.

[7] L. Blunk, M. Karir, and C. Labovitz. 2011. Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format. RFC 6396 (Proposed Standard). (Oct. 2011). DOI: https://doi.org/10.17487/RFC6396

[8] Dyn. 2010. China's 18-Minute Mystery. (nov 2010). <http://dyn.com/blog/chinas-18-minute-mystery/>.

[9] Dyn. 2014. Indonesia Hijacks the World. (apr 2014). http://dyn.com/blog/indonesia-hijacks-world/.

[10] endolith. 2011. Peak Detection MatLab Algorithm. <https://gist.github.com/endolith/250860>. (2011). Visited June 2017.

[11] Jared Mauch. 2007. Detecting Routing Leaks by Counting. Detecting Routing Leaks by Counting. (2007).

[12] Qing Ju, Varun Khare, and Beichuan Zhang. 2010. Large Route Leak Detection. (jun 2010). <https://www.nanog.org/meetings/nanog49/presentations/Tuesday/LRL-NANOG49.pdf>.

[13] M. Lepinski and S. Kent. 2012. An Infrastructure to Support Secure Internet Routing. RFC 6480 (Informational). (Feb. 2012). DOI: https://doi.org/10.17487/RFC6480

[14] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. 2013. BGP Prefix Origin Validation. RFC 6811 (Proposed Standard). (Jan. 2013). DOI: https://doi.org/10.17487/RFC6811

[15] Lucas Hermann Negri. 2017. Peak Utils. http://pythonhosted.org/PeakUtils/. (2017). Visited June 2017.

[16] Ricardo Oliveira. 2009. Cyclops Open eye to your network. (jun 2009). <https://www.nanog.org/meetings/nanog46/agenda>.

[17] Y. Rekhter, T. Li, and S. Hares. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard). (Jan. 2006). DOI: https://doi.org/10.17487/RFC4271 Updated by RFCs 6286, 6608, 6793, 7606, 7607, 7705.

[18] RIPE-NCC. 2017. Routing Information Service (RIS). <http://www.ripe.net/data-tools/stats/ris/>. (2017). Visited June 2017.

[19] Scipy. 2017. find_peaks_cwt. <https://docs.scipy.org/doc/scipy-0.19.0/reference/generated/scipy.signal.find_peaks_cwt.html>. (2017). Visited June 2017.

[20] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, and B. Dickson. 2016. Problem Definition and Classification of BGP Route Leaks. RFC 7908 (Informational). (June 2016). DOI: https://doi.org/10.17487/RFC7908

[21] ThousandEyes. 2014. Proactive BGP Alerting. (nov 2014). <https://blog.thousandeyes.com/proactive-bgp-alerting/>.

[22] Valadon, Guillaume and Vivet, Nicolas. 2014. Detecting BGP hijacks in 2014. (Nov. 2014).