Poster: Evaluating RPKI ROV Identification Methodologies in Automatically Generated Mininet Topologies

Nils Rodday
Research Institute CODE

Ruben van Baaren
Radboud University

Luuk Hendriks
NLnet Labs

Roland van Rijswijk-Deij
University of Twente and NLnet Labs

Aiko Pras
University of Twente

Gabi Dreo
Research Institute CODE

ABSTRACT

The deployment of the Resource Public Key Infrastructure (RPKI) is currently gaining traction within the operator community and so are measurement methodologies trying to measure the current deployment status. These methodologies, which are attempting to infer Autonomous Systems (ASs) performing Route Origin Validation (ROV), are applied onto real world data but validation of the results is usually hard as no ground-truth dataset exists. We propose to build such a dataset with the help of Mininet in a way that ROV measurement methodologies can be evaluated within a testbed in which filtering ASs are known to the experimenter. The Mininet topology generator will not only be specific for RPKI measurements but could be used also as an evaluation testbed for any kind of measurements that require a ground-truth dataset. Our framework is fed with real world BGP collector data, renders an abstraction of the acquired topology graph and translates it into a Mininet topology. In our scenario, RPKI filtering is deployed within the topology such that the testbed can later on be used to evaluate existing RPKI ROV measurement methodologies. We therefore plan to contribute an automated Mininet topology generator and insights into the accuracy of current ROV identification methodologies.

ACM Reference Format:

Nils Rodday, Ruben van Baaren, Luuk Hendriks, Roland van Rijswijk-Deij, Aiko Pras, and Gabi Dreo. 2020. Poster: Evaluating RPKI ROV Identification Methodologies in Automatically Generated Mininet Topologies. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3386367.3431669

1 INTRODUCTION

Currently, researchers performing simulation-based studies of the BGP ecosystem need to develop their own topologies within Mininet for the evaluation of each approach and deploy their measurement specific software in the testbed in order to obtain a ground-truth dataset. This is error-prone and requires a substantial amount of time and effort to build. Therefore, validation using a ground-truth dataset is rarely performed. Instead, pointers within the real world dataset are used to argue for a working approach. This work proposes a framework that automatically translates graphs obtained from publicly accessible BGP collector projects such as Routeviews and RIPE RIS, into working Mininet topologies to simplify the evaluation within a simulated network topology. To this extent, we perform the following steps: (i) Collect BGP measurement data from Routeviews and RIPE RIS, (ii) translate those datasets into a directed graph in which ASs are represented by nodes and relationships between ASs by directed edges, (iii) abstract that large graph into a smaller version while still maintaining the degree distribution of nodes, (iv) automatically generate BGP daemon configuration files based on the abstracted graph.

Our use case is RPKI Route Origin Validation (ROV) measurement studies. Many exist but rarely have they been compared within the same set of variables. Instead, vantange points, location of prefix announcements, and other variables differ. Hence, it is hard to say which approach performs better compared to the other. Our framework aims at solving this problem by evaluating the approaches within the same set of boundaries in a Mininet topology.

2 MININET TOPOLOGY CREATION

We start by extracting paths from publicly available BGP collector projects using BGPReader [6], depicted as step one in Figure 1. It provides a simple Python interface to access data from Routeviews and RIPE RIS. For our preliminary results we use an one-hour input dataset from September 28th, 2020 2 am - 2:59 am of all available collectors. The dataset has a size of 6.4 GB and contains ~44 M BGP update messages. We are able to extract 39685 ASs, which are represented later on as nodes in our graph. Additionally, we extract the links between ASs by looking at the AS_PATH BGP attribute. If, e.g., the path contains "AS1 AS2 AS3", we infer relationships between AS1 and AS2, as well as, AS2 and AS3. In order to properly add those relationships to our graph, we utilize the CAIDA AS relationship dataset [1] from September 1st, 2020, illustrated as step two in Figure 1. We will add two types of links: directed and undirected. While the former will represent a customer $\rightarrow$ provider relationship, the latter will be added for peer $\leftrightarrow$ peer connections. If a link cannot be found in the CAIDA AS relationship dataset, we discard such link. Adding edges without relationship information will pollute the graph as the type of link determines at a later stage how the BGP daemon configuration files are built. Therefore, the user will have to consider a tradeoff between quality (correct links) and quantity (amount of links). We plan to allow to enable/disable this feature and leave it up to the user's discretion which argument is deemed more important, depending on which experiment the Mininet generator will be used for. Out of 148.4 M links, 76.7 M

CoNEXT '20, December 1–4, 2020, Barcelona, Spain

ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431669

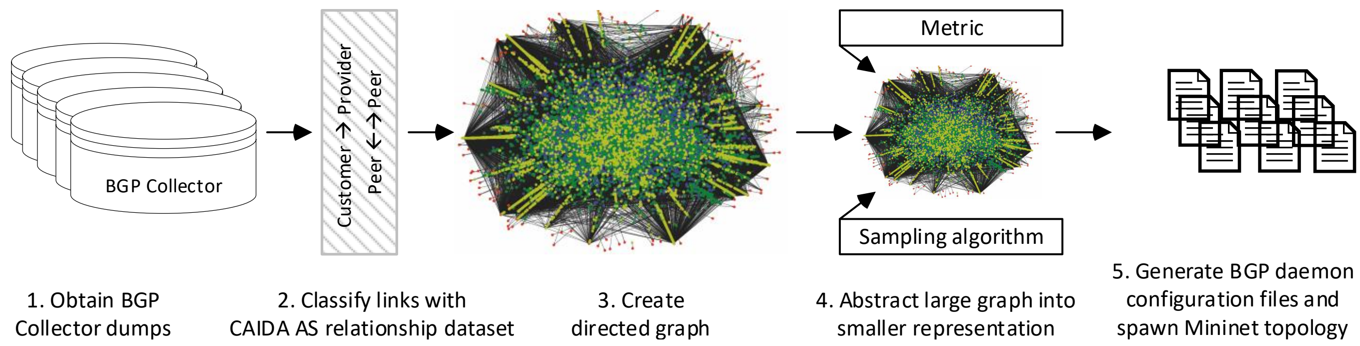

Figure 1: Methodology

(51.7 %) could be found. Since many BGP updates contain the same links and are therefore duplicated, the resulting graph has 59585 edges. With the extracted data a graph representation is created in NetworkX [5], a Python based library to study complex networks as graphs. This is shown as step three in Figure 1. Next, we eliminate all isolated nodes within the graph, in our case 2175 nodes, and identify the largest subgraph in the entire representation. All other subgraphs are removed (as those are not connected), which eliminates another 137 nodes. The remaining graph contains 37373 nodes and 59414 relationships.

Given the architecture of Mininet as well as memory and processing capacities required for each node, a simulation of the entire graph is currently not feasible. Therefore, a crucial step is performed next, when an abstraction of the graph is created to shrink it to ~4000 nodes, depicted as step four in Figure 1. It is very important to maintain as many features of the original graph as possible such that conclusions drawn from the abstraction can be generalized and applied to the real world. First, we need to define a metric that will indicate how well a sampling algorithm worked for the use case. For our RPKI ROV use case, we chose the degree distribution of nodes to be the most crucial metric. The results of the upcoming study will strongly depend on this criterion. E.g. if large transits filter RPKI invalid route announcements, some methodologies might falsely attribute this filtering to another leaf AS as it is "behind" the transit, instead of the transit itself. Second, we need to choose a sampling algorithm. The closer the predefined metric is in the original graph compared to the abstracted graph, the better the sampling algorithm worked. To perform the sampling, we use the graph sampling library LittleBallOfFur [7] and generate our abstracted graph applying the RandomWalk sampling algorithm. Since our framework is designed in a modular fashion, the abstraction algorithm could be replaced with any other algorithm provided by the aforementioned sampling library or any self-written algorithm, giving the experimenter the freedom to decide which features are most important to preserve in the abstraction. Also the metric could be redefined for each specific use case, making it easy for a user to find the best sampling algorithm that preserves most features.

Once a smaller abstraction has been obtained we aim at the automation of a Mininet topology creation based on the abstracted, directed graph, pictured as step five in Figure 1. As we are using the Bird BGP daemon [3] within our Mininet infrastructure, Bird configuration files are generated to allow for BGP peering sessions within the emulation. However, the generator could be extended with other BGP daemon output formats, allowing a user to generate topologies to be run with different BGP daemons. The generation process respects the Gao-Rexford model [4] by only exporting routes according to the conditions from [2]. We distinguish the following cases accordingly: exporting to a provider, customer, peer, or sibling.

Finally, RPKI ROV is randomly deployed on a user-defined share of nodes. It is therefore known which ASs filter in the simulation environment for each run. By applying existing ROV measurement methodologies within the testbed we will be able to judge on their accuracy regarding identification.

3 CONCLUSION

This work presented a framework that is able to (i) extract ASs and AS relationships from BGP collector dumps, (ii) create a directed graph representing the topology, (iii) perform an abstraction into a smaller graph using a metric and a sampling algorithm of the user's choice, and will be able to (iv) translate this into BGP daemon configuration files. The resulting Mininet topology is intended to be used as an evaluation testbed for BGP related experiments. While we plan to evaluate RPKI ROV measurement methodologies within the testbed the topology generator itself is not limited to this scenario and might prove useful for many other experiments as well.

REFERENCES

[1] The CAIDA AS Relationships Dataset. 2012. https://www.caida.org/data/as-relationships/.

[2] Xenofontas Dimitropoulos and George Riley. 2006. Modeling autonomous-system relationships. In *20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)*. IEEE, 143–149.

[3] Ondrej Filip, L Forst, P Machek, M Mares, and O Zajicek. 2010. BIRD internet routing daemon. *NANOG-48, Austin, TX* (2010).

[4] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM Transactions on networking* 9, 6 (2001), 681–692.

[5] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[6] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. 2016. BGPStream: a software framework for live and historical BGP data analysis. In *Proceedings of the 2016 Internet Measurement Conference*. 429–444.

[7] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. 2020. Little Ball of Fur: A Python Library for Graph Sampling. arXiv:2006.04311 [cs.SI]