Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Resilient Networks

Exercise Sheet: Routing

Prof. Dr. Mathias Fischer, August See

Goals and Objectives Recapitulate the lecture on routing and then apply this knowledge to investigate the usage of BGP on today's Internet.

Tools and Data The RIPE Network Coordination Centre (NCC)[1] with their Routing Information Service (RIS)[2] collects BGP messages for example at Internet Exchange Points (IXP) and make them available to the public. This works as follows: At the respective IXPs, the RIS operates own BGP speakers that BGP speakers of other ASes can connect to, i.e., peer to. Consequently, peered ASes send BGP messages to the BGP speaker of the RIS to propagate routing information. The RIS monitors their BGP speakers and provide the BGP information as an export from the respective BGP router software. For this exercise, we are using the collection of BGP messages at the DE-CIX in Frankfurt, Germany[3]. The dataset for this exercise is a snapshot taken on November 6th in 2019. This BGP export is available for download at:

http://data.ris.ripe.net/rrc12/2019.11/bview.20191106.1600.gz

We recommend to make use of the tool *bgpdump*[4] to output BGP information on the commandline, and to use the python wrapper *bgpdumpy*[5] for more detailed BGP analysis. On your Linux VM, you can prepare the tools by executing:

```
$ cd ~/Desktop
$ wget https://svs.informatik.uni-hamburg.de/teaching/rn/exercise/03/routing_setup.sh
$ chmod +x ./routing_setup.sh
$ ./routing_setup.sh
```

As usual, you are not limited to the recommended tools but can use any tool or programming language of your choice.

[1] https://www.ripe.net
[2] https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris
[3] http://data.ris.ripe.net/rrc12/
[4] https://github.com/RIPE-NCC/bgpdump
[5] https://github.com/AlexForster/bgpdumpy

Submission Solve this task sheet in groups as you organized yourselves beforehand. Your solution should be handed in as a pdf document that describes your results and how you got there. In particular, your solution should document the following:

- A description of your approach and methodology towards the solution.
- A summary and interpretation of your results (figures, tables, examples. . .).
- The code that solves the tasks.

Submit your group solution to this task sheet in moodle.

1 Global Routing Tables

The RIPE RIS collects all the BGP update messages from peered ASes that participate in this service. Thus, the RIS retrieves BGP information (announcements and withdraws) from several peered ASes all the time, effectively reflecting their routes. Based on their streams of BGP messages, the RIS publishes snapshots of valid routes at the specific snapshot time[6]. Make sure you downloaded and use the export given above in **Tools and Data**.

1.1 Peers and Routes in the Export

Find out about the ASes that peered to the RIPE RIS (aka. those peers that are **directly connected** to RIS) and calculate statistics about their BGP activity, e.g., by using the tool *bgpdump* and some well-known Linux commandline tools such as *grep*, *sort* and *uniq* to process the output.

1. Look at the output of *bgpdump* and specifically at the structure of a route entry. Which key word indicates the peered AS from which the entry was received?
2. Routes from how many different peered ASes, i.e., BGP speakers, are contained in this export? **Hint**: Pipe the `bgpdump` output to the `grep` command to filter for the respective key word, pipe it to the `sort` command to sort the result, pipe it to the `uniq` command to eliminate duplicates, and return the number of unique BGP peers by piping it to `wc -l`.
3. How large (min and max) are the routing tables of the peered BGP speakers? Calculate the number of routes received from each speaker. **Hint**: You can use a similar chain of commands, but this time use `uniq -c` to count occurrences in the sorted list

[6] For an example for BGP messages (instead of a snapshot), see the expert task 4

1.2 Extracting a Specific Routing Table

For the remainder of this task sheet, we want to look at the IPv4 routes of a particular AS (`AS64475`). Thus, reconstruct the routing table of this AS for the given snapshot and prepare for the analysis of its BGP routes.

You can make use of our example python file `routing_example.py` (find it in moodle) that provides you with some basic program structure. On your prepared Linux VM, run the script as follows (assuming the BGP export is in the same folder):

```
$ cd ~/Desktop
$ wget https://svs.informatik.uni-hamburg.de/teaching/rn/exercise/03/routing_example.py
$ python3 ./routing_example.py bview.20191106.1600.gz
```

Alternatively, checkout the `-M` flag of `bgpdump` and process the output in any way you feel comfortable.

1. Explain how the filtering for `AS64475` works when browsing the dataset, e.g., how is our example python file achieving this.
2. Using your detailed BGP analysis, extend you program to count the number of routes received via `AS64475`. Verify that it is the same number you calculated for question 3. in Task 1.1.

2 Analyzing BGP Routes

In the lecture, you already learned how routes propagate on the Internet among ASes. New routes are advertised by a vector of AS numbers, i.e., the AS path. Continue to use your program for detailed BGP analysis from Task 1.2, e.g., the example python file that we provide. Analyze the following routing aspects of `AS64475`.

2.1 Characterization of AS64475

First characterize the `AS64475` itself, who this AS actually is and which IP prefixes this AS owns.

1. Use a commandline tool such as `whois` to find out who is the organization behind this AS.
2. On the basis of the routes in the BGP dump, how many IP prefixes indicates the `AS64475` to own. When looking at these prefixes, do all seem reasonable? **Hint**: You can identify all routes with an empty AS path.
3. On the basis of your sanitized list of owned prefixes, which fraction of the overall IP address space does this AS own?

2.2 Characterization of Routes

Next, characterize the routing table of `AS64475`. In contrast to Task 2.1, for the following questions, look at only the *foreign* IP prefixes, i.e., all routes to prefixes that are not owned by this AS.

1. How many routes to foreign IP prefixes exist in the routing table?
2. What is their average AS path length?
3. How large is the largest announced IP block in terms of network size? How many IP addresses are available in such a prefix?

Furthermore, analyze the ASNs in the AS paths to learn about other ASes on the Internet.

4. How many unique ASNs exists among all the routes of `AS64475`?
5. How many ASes originate own IP prefixes? How many ASes just forward traffic as a transit AS?

3 Resilience of Internet Connectivity

Based on the route analysis of Task 2.2, evaluate the routes of `AS64475` regarding the resilience of its Internet connectivity against the failure of other ASes. Similar to the graph analysis in the first task sheet, imagine this routing table to spawn a tree-graph that is rooted in `AS64475`, where the leaf nodes reflect IP prefixes and the intermediate nodes reflect other ASes on the path towards the respective IP prefix. Hence, the graph represents the view of `AS64475` on the Internet.

1. How many first hops exist in this graph, i.e., via how many directly neighbored ASes do routes exist?
2. Which is the most critical AS for providing Internet connectivity in terms of frequency? Identify the AS that most often occurs among all routes.
3. Which is the most critical AS for providing Internet connectivity in terms of network size? Identify the AS via which the largest number of IP addresses can be reached.
4. The connectivity to which fraction of the overall IP address space would potentially be lost when this critical AS becomes unavailable?
5. Why would the consequences of such a failure be more relaxed in practice?

4 Optional: BGP Messages

In this optional expert task, you are going to analyze BGP update messages, not a snapshot of the BGP routing table. We are using the same data source for this purpose. However, this time we refer to the export of messages between 12:40 and 12:45 on November, 6th in 2019. The respective file can be downloaded here:

http://data.ris.ripe.net/rrc12/2019.11/updates.20191106.1240.gz

You can also use the tool *bgpdump* with the flags `-m` or `-M` to parse the export of BGP messages and to write the result to a file. Unfortunately, the python wrapper *bgpdumpy* is not capable of processing the dump of messages, so you probably want to process the *bgpdump* output in any way you prefer.

4.1 Characterizing the Route Graph

For all paths towards an IP version 4 (IPv4) destination, create a route graph according to following rules:

- Each AS is a node in the graph.
- If an AS is originating one or more destinations, the aggregated subnet information is linked to the respective node.
- Directed edges between the ASes represent the paths of a route towards a IPv4 destination.

Based on this graph, together with the IPv4 destinations, answer the following questions:

1. How many nodes, i.e. ASes, are in the graph?
2. How many percent of the ASes originate an own IP address space?
3. What is the average shortest distance and the diameter of the graph?
4. What is the minimum, average, and maximum size of the IP address spaces among the ASes?
5. To which fraction of the full IPv4 address space would you been able to route packages? For that, accumulate all IP address spaces.

4.2 Detecting BGP Attacks

Now that you gained so much experience in analyzing BGP messages, you are going to implement some basic measures to detect potentially malicious BGP speakers for the IPv4 address spaces.

1. Receiving the same route (identical AS path and destination) multiple times via the same neighbored BGP speaker indicates route flapping - frequent ups and downs of routes. Find out the top 10 routes that show this behavior by looking for duplicate advertisements.

2. Usually ASes advertise their whole address space in big blocks for BGP routes. Hijacking IP addresses is possible when attackers advertise routes with longer prefixes. For a simple detection, identify all ASes that advertise routes to a destination with a network length of 30 bits or more. Use tools like *whois* to find out the *actual* AS where these long-prefix destinations belong to. Can you find a relation to the potentially *malicious* AS.

3. More generally, public IP addresses are supposed to be globally routed to a single destination or AS, respectively. Thus, identify ASes that originate overlapping IP subnets.