GaN Network Lab 1

Introduction

The goal of this first lab is to run simple measurements such as Ping, Traceroute and DNS on the RIPE Atlas platform in order to find information about Content Delivery Networks (CDNs) infrastructure.

Learning objectives:

- Learn how to retrieve measurement information on RIPE Atlas
- Learn how to perform measurements on RIPE Atlas
- Understand what a CDN is
- Understand how users are redirected by CDNs

Content:

- Setup working environment
- Use RIPE Atlas User Interface (UI)
- Use RIPE Atlas API
- Retrieve measurement and IP address geolocation
- Perform your own measurements
    - Ping: latency evaluation
    - Traceroute: IP/AS level path

Setup environment

For this first lab, we will be using Jupyter notebooks and pythons scripts to run measurements and retrieve data using RIPE Atlas API. We will also set up a simple python environment to install additional packages.

Requisites:

- Python >= 3.10 (python installation)

```
python3 --version
pip --version
```

- Git (git installation)

```
git --version
```

- Jupyter (jupyter installation)

```
pip install jupyter ipykernel
jupyter --version
```

Once all the dependencies are installed, install the default lab template by following the next steps.

Step 1: Open a terminal and create a fresh directory to put all your code:

```
mkdir -p ./GaN/network_labs
cd ./GaN/network_labs
```

Step 2: Clone the GitHub repository

```
git clone https://github.com/hrimlinger/GaN-network-labs.git
cd GaN-network-labs
```

This repository does not contain a lot…for now! We will add things along the way.

Step 3: Create a python virtual environment, activate it and install the necessary packages

```
python3 -m venv .GaN-venv
source .GaN-venv/bin/activate
```

Step 4: Install dependencies using the **requirements.txt** file:

```
python3 -m pip3 install requirements.txt
```

You are now within a so-called *python virtual environment*. This means that a dedicated python interpreter is attached to your project. You can install any dependencies you want without risking entering in conflict with another user/project installation. This is particularly handy when working on multiple projects or when sharing the same machine with other users (as it is the case for the lab's computers).

Finally open the lab's jupyter notebook either via your favorite IDE (Integrated Development Environment), i.e. Visual Studio Code, PyCharm, etc… or by running:

```
python -m ipykernel install --user --name=.GaN-venv
--display-name=".GaN-venv"
jupyter notebook lab1.ipynb
```

In the notebook, do not forget to select the right python kernel (in our case **.GaN-venv**)
At this point, you should be able to execute python code within a jupyter notebook. Take a few minutes to read the introduction about CDNs and RIPE Atlas before starting answering the question.

Content delivery networks

CDN infrastructure illustration from https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

What is a CDN? CDNs are large geographically distributed networks providing high performance content distribution.. A CDN relies on a geographically diverse network infrastructure of cache servers to efficiently serve end-users all over the world. While ensuring high performance to their users, CDNs also provide security mechanisms, reduced bandwidth cost and increased website availability. There is much more to say about these networks but this CloudFlare blogpost offers a short and clear introduction to CDNs.

Why is it interesting to study them? As you may expect, CDNs have a central role in today's Internet exchanges and are responsible for most of the carried traffic. They shape Internet usage and technologies and any person with a working Internet connection has interacted with one of them.

Goal for this lab. In this lab, our goal is to better understand the different strategies adopted by these CDNs to redirect their clients to an optimal Point-of-Presence (PoP, i.e. an entry point, a frontend-server). More specifically, we will study two CDNs (Google and CloudFlare) with very different approaches to handle the problem of client redirection.

Challenge and limitation. Measuring the performances a user can experience while requesting a service from a CDN is challenging because of the distributed nature of these infrastructures. Thankfully, non-profit organizations allow Internet Service Providers (ISP), researchers and students to run measurements from geographically diverse servers (called **vantage points or probes**). As you will discover: RIPE Atlas is slow. Slow to perform measurements, slow to upload the results, etc…Or should we say, not fast enough to study CDNs infrastructure globally. For that, different tools and strategies are necessary.

RIPE Atlas measurement platform

RIPE Atlas vantage points

What is RIPE Atlas? RIPE atlas, with its 10k probes (or vantage points), in 177 countries and 3690 ASNs, is the current largest publicly available measurement platform. It is used by everyone, both operators and academics, to retrieve, test, measure and troubleshoot the Internet. RIPE Atlas is a collaborative measurement platform where every user can host a vantage point, earn credits and run measurements. Nevertheless, the results of millions of daily measurements are freely available for download. Because the probes are hosted in countries with different laws (especially in terms of Internet censorship) only a few types of measurements are available: Ping, Traceroute, DNS and TLS. We will focus on the first three.

RIPE Atlas User Interface (UI). RIPE Atlas website provides a convenient way to consult the result of a measurement, find information about a vantage point (its network, geographic location, status, etc.).

RIPE Application programming interface (API). RIPE Atlas also exposes a programmable interface (or API) to allow measurement automation with code. An API serves as a software to provide secure and comprehensive access between users and a complex application (here the measurement platform). With the help of GET/POST requests to endpoints provided by the RIPE Atlas API you will be able to retrieve and start measurements.

Running measurement. While retrieving data is free, running measurements cost credits. You will be provided a **SECRET_API_KEY** to authenticate on our research account. This key must be **KEPT PRIVATE** at all times as we are using it to run our own measurements. If you are unsure about the measurement you want to start, please ask your lab teacher for confirmation. All measurements will be monitored to avoid any problems (do not worry too much, the API will tell if something goes wrong).

Get measurement from RIPE Atlas UI

In this section, we will retrieve measurement data and analyse them, first using the UI and then using python code to make requests to the API:

- UI
- API

Using the UI

Step 1: Go to the UI and navigate to the measurement endpoint (on the left scroll bar).

Step 2: In the search bar enter the measurement id: **83506365.**

1. What type is the measurement? (Explain briefly how this measurement works)
2. What kind of information can you retrieve with this type of measurement?
3. Which IP addresses are targeted?
4. When was it created? Finished?
5. What information can you find from the measurement description?
6. How many vantage points were requested?
7. How many vantage points participated?
8. How many packets were sent per vantage point?
9. Check the "results" entry of the measurement page:
    a. Where is located the vantage point with the lowest latency?
    b. What is its public IP address? IP prefix? AS number?
    c. There exists a correlation between the geographic distance and the measured latency, can you infer the location of the IP address?

Step 3: In your terminal, run the following command (replacing the IP address with the one of the measurement):

```
whois <ip_addr>
```

1. What is whois? (briefly)
2. What is the prefix of this IP address?
3. Who owns this prefix?

Without answering all the previous questions, check the measurement **82979479.** Where is the measured IP address located? What does it say about this CDN infrastructure?

Using the API

Open the notebook **lab1.ipynb.** In this notebook, you will find a small example to retrieve a past measurement from the RIPE Atlas platform.

Having access to the result of a measurement via a user interface is nice, but it does not scale. For this reason, RIPE Atlas also proposes an API to retrieve/perform measurement on the platform. An API is a program in charge of simplifying the connection between a user and an application. APIs are

everywhere: in the Linux kernel (or any other OS), to write/read from a storage device, Web API, to post pictures/videos, etc. In our case, we will use two functions of the RIPE Atlas API:

- **GET:** Access information of a measurement from the database
- **POST:** Ask for a measurement to be performed

Notice that you can also **DELETE** or **UPDATE** measurements, as any REST API.

This section is separated into two:

1. Retrieve the measurement information (number of probes requested, creation time, description, etc.)
2. Retrieve the measurement results.

Step 1: Open the notebook **lab1.ipynb.** In it, you will find the default structure for executing measurement and retrieve the data. Follow the steps described in the notebook, complete the exercise and answer the question using the retrieved results.

Exercise 1:

1. Get the measurement information
2. Answer the same question of the previous section (creation time, number of probes, etc.)
3. Get the measurement results
4. For each participating probe, print: '**source_addr'; 'from'; 'prb_id'; 'dst_addr'; rtts; min_rtt** where min_rtt is the minimum rtt over all the packet sent.
    a. What is problematic with the field **'src_addr'**?
    b. For a few vantage points, use the RIPE Atlas probe UI and get their network information. What is the **'from'** parameter?
5. Order the vantage point measurements per **increasing min_rtt** and find the vantage point with the lowest latency. Where is it located?
6. Knowing that the maximum speed of a packet through the Internet is ⅔ of the speed of light (speed of a packet in optic fiber), estimate the distance between the vantage point with the lowest latency and the target IP address. Geographically speaking, what is the shape of the *"area of presence"* of the target around the VP?
7. What are the limitations of geolocating an IP address with this technique? Why is it often imprecise?

Exercise 2:

8. Perform a ping measurement to IP address **142.250.201.36** (this time using a POST request, example given in the notebook).
9. Get the measurement results
10. For each VP, get the **min_rtt** (as previous)
11. Check the two VPs location, compare the measured latencies. Why is a VP experiencing a much higher latency? (note: to help you, check the location of the target IP address using IPinfo)

Exercise 3:

1. Do the same but for IP address: **104.16.124.96**
2. Do you observe a difference with the previous latency measurement?

Exercise 4:

1. Run Traceroute measurements from the same VPs targeting the two IP addresses and print the IP level path between the destination (the VP) and the target IP address (either 1 or 2).
2. In your opinion, do you think these measurements reflect the reality a user could experience while requesting services to those CDNs?

We will leave the rest of this analysis for the next lab. You will also have a complete CM about Traceroute so, to avoid overlapping, simply understand that Traceroute is a tool to measure the path taken by packets on the network between a source (the vantage point) and a destination (in our case the CDN's frontend-server).

Bibliography:

RIPE Atlas:

- UI
- API
- RIPE RIS
- RIPE NCC
- RIPE Atlas survey
- RIPE Atlas IP address geolocation

CDNs:

- CDNs resiliency paper
- CloudFlare CDN
- Google CDN

Others:

- IPInfo
- PeeringDB

Anycast:

- CloudFlare Anycast
- Manycast (Anycast detection paper)