Architecture

This document describes the architecture of the MANRS benchmarking tool. Each file in the tool is well-documented and reading the documentation strings will provide all the technical details that this document may lack.

The MANRS benchmarking tool gathers data from public sources and calculates metrics on a given set of ASNs. For now the given set of ASNs is gathered from the MANRS participants page. The `get_manrs_participants()` function in `manrs/util.py` is used for that.

The tool consists of the following main parts:

- Benchmark
- Data sources
- Metrics
- Database
- API

Benchmark

`benchmark.py` is the core application of the tool. When run it will:

- Fetch the MANRS participants;
- Initialize and use the data sources' modules;
- Use the data sources' results to calculate metrics;
- Create the report and store the data along with statistics in the DB.

By default it gathers and parses data for the *previous* month. Ideally it should be run on the first day of each month.

`benchmark.py -h` lists the available options that can be used to generate reports for specific periods.

Data sources

Data sources provide their own logic for obtaining data from public data sources. These can be found as individual modules under the `manrs/data_sources` directory. The output of these modules influences heavily the way Metrics are being calculated and the way Benchmark creates and stores the results in the Database. Thus changing or adding new data sources requires changes to these parts as well.

BGPstream

The `bgpstream.py` data source is responsible for fetching and parsing data from the BGP Stream service. The type of available information is BGP leak and BGP hijack events. It requests and parses all events during the testing period in a single HTTP call.

There are two outcomes per event: culprit and accomplice. Culprit is the ASN that started the event (i.e. leaker, hijacker). Accomplice is the next-hop ASN found in the AS-PATH.

CIDR report

The `cidr.py` data source is responsible for parsing data gathered locally from the CIDR report service. The type of interesting information is bogon prefix and ASN advertisements. Currently only culprits for bogus IPv4 prefix advertisements are parsed.

Daily acquisition of data

Because of the non-historic data nature of the CIDR report, we need to fetch the available data daily and store it in a date-hierarchical folder structure. The python script `cidr/get_daily_cidr.py` should be ran daily in order to store the data in the `cidr/data` directory. It should be ran in a sane interval (ie. every three hours) to check for date changes in the CIDR website. When the CIDR report is updated, the new data is fetched and stored. If the data is already stored for the given day the script has nothing more to do. For now it stores the file `bogon_prefixes.txt` with bogon IPv4 prefix advertisements.

Ripestat

The `ripestat.py` data source is responsible for fetching and parsing data from the RIPEstat service. RIPEstat offers a lot of information on an ASN but we are currently interested on the `AS routing consistency` and the `Whois` API

data calls. The former provides information on the advertised and registered prefixes while the latter provides information on the registered contact details.

We need to issue an HTTP request for each data call and each ASN. The HTTP calls are made asynchronously respecting RIPEstat's parallel limit (8). However, RIPEstat is currenlty running on full capacity and some requests for ASNs with considerable amount of prefixes that require a lot of memory to generate results may fail.

Adding new data sources

New data sources can be added in the `manrs/data_sources` directory. There needs to be logic for fetching and parsing the required data. Note that data sources and metrics have no immediate connection so an extra data source can be used to combine data with another data source in order to calclulate a metric. Additionally, further changes are required in the following files in order to use the new data sources:

- `benchmark.py`, where the data sources are initialised and used;
- `manrs/models.py`, if we need to change the DB schema in order to store new data;
- `manrs/metrics.py`, if we need to introduce new metrics for the data source or update existing ones.

Metrics

Documentation on metrics can be found in the `doc/Metrics.{rst, pdf}` files. Metrics are calculated in `manrs/metrics.py`. Each metric requires specific data from specific data sources. Currently each metric relies on a single data source but because the data source and metric logic is decoupled it would be possible to combine data sources for a single metric's calculation.

Database

The database used for the tool is PostgreSQL. Interfacing with the database is accomplished throught SQLAlchemy's ORM system <https://docs.sqlalchemy.org/en/latest/orm/>___. The schema for the database is defined in manrs/models.py.

The provided create_db.py‘ python script will create the schema the first time it is run and update the schema in consecutive runs if the `manrs/models.py` file was updated.

API

The API is defined in the `api.py` file. It uses Falcon, a web API framework for Python.

Complete documetation on the API functionality is available in the `doc/API.{rst, pdf}` files.