Greenwich Meanie Documentation

1 Overview

This code analyzes the data associated with Greenwich meanie. Greenwich meanie is characterized by:
- Many source addresses sending packets to the same high-numbered UDP port
- Ports look random or encrypted, between 49152 and 65535
- Most source addresses send a couple packets, then disappear
- Packets contain random or encrypted data, varying between 93 and 227 bytes
- Port number changes daily at midnight GMT
- First observed in February 2020

There are two pieces of the code:
- Code that analyzes the packets associated with Greenwich meanie data
- Optional code: Analyzes geolocation information, and/or device types for the meanie source IPs

2 Dependencies

The dependencies files (for python 3.10.4) are located in the ***requirements*** folder in the code location

2.1 Pip Install

Included requirements_pip.txt will install necessary python libraries. Use the following command:

```
pip install -r requirements.txt
```

2.2 Manual Install

You can also choose to install manually. Table 1 shows the required python libraries.

Required Dependencies	Required Dependencies for Optional Code
python=3.10.4	requests=2.31.0
matplotlib=3.5.2	json5=0.9.14
numpy=1.25.2	ipinfo=5.0.0
pandas=2.0.3	urllib3=2.1.0
pip=22.1.2	
geopandas=0.13.2	
openpyxl= 3.1.2	

Table 1: Code Dependencies

3 Input Data Description

The Greenwich meanie code takes as input .txt files of Greenwich meanie data.

The code takes a directory as an input and analyzes all the .txt files in there. The .txt files can be generated using the code in the meanies folder. We assume that you have already run that step. If you want to analyze a subset of the data, you have to do that beforehand by only adding the appropriate files to the input directory. Data requirements for the code:

- For running the yearly analysis: Have a subsample of data from each year to be analyzed. It could be any amount of data for that year. For our experiments, we sampled data for 2nd Wednesday of every month.
- For the weekly and daily analysis: at least one whole week is needed for each year. We sampled 1 week for each of the 4 years analyzed.

The information below on file format is not necessary for the users to know as long as they provide the correct input files, but documented here for reference. The code handles all this information internally. The meanie text files contain a packet in each row containing the following comma-separated values (in order):

1. Source Address: Stored as a hex value
2. Destination Address: Stored as a hex value
3. Source Port: Stored as a hex value
4. Destination Port: Stored as a hex value
5. Protocol: Stored as decimal value and always 17 since meanie activity only happens on UDP
6. Timestamp: Stored as Julian
7. UDP Check Sum: Stored as a hex value
8. Payload Length: the length of the payload, in bytes (the UDP length field, minus 8 for the UDP header itself)
9. Payload: Stored as a hex value
10. TTL: Stored as a hex value
11. IPID: Stored as a hex value

4 Pre-requisite Inputs from External Modules for Optional Code

There are 2 pieces of optional code: i) Finding geolocations of the IP addresses or ii) finding the device types associated with the IP addresses. Users can run one or both these pieces

4.1 Geolocation Module

Ipinfo.io extracts geolocations and autonomous systems (ASNs) for IP addresses. Details:

1. Create an account on ipinfo.io and add your key(s) to the code config file
2. Ipinfo.io allows querying information for only 50,000 IP addresses per month with the free account but you can pay for more IPs and more information.

Table 2 shows the sample geolocations information that the code extracts.

IP	City	Region	Country	Longitude	Latitude	AS_Number	AS_Name	Timestamp
103.16.13.156	Karur	Tamil Nadu	India	78.0810	10.9577	132556	Blue Lotus Support Services Pvt Ltd	10/12/2023
103.221.255.242	Dhaka	Dhaka	Bangladesh	90.4074	23.7104	135524	university of dhaka	10/12/2023
49.37.37.36	Kolkata	West Bengal	India	88.3917	22.5333	55836	Reliance Jio Infocomm Limited	10/12/2023
93.136.62.203	Pula	Istria	Croatia	13.8481	44.8683	5391	Hrvatski Telekom d.d.	10/12/2023

Table 2: Example of data in geolocations database

4.2 Device Types Module

Lift (https://github.com/trylinux/lift) returns device types associated with each IP address. Details:

1. So far, lift has returned results for only 0.77% of the total IPs queried. So running lift is optional but may be useful for different set of IPs
2. When running lift, remember to save the output to a text file that can be processed by the code.
3. Lift only works in Linux and using virtual machine is fine.
4. Lift website gives straightforward instructions for downloading, installing, and running
5. Lift runs slowly so using concurrency parameter (described in lift documentation) helps speed things up, but even then it is fairly slow.

Table 3 shows sample output of the code that processes lift results. It classifies each device to one of the following device types: Server, Router, Recorder, Camera, NVR, HVR, or DVR.

Original_Device_Info	Timestamp	IP	Port	Device_Type
2023-10-11 17:08:25.751818 | 85.242.215.127:80 | has server None and no viewable title (NOID)	10/11/2023	85.242.215.127	80	Server
2023-10-11 17:44:23.023562 | 103.5.132.3:80 | ZTE MA-DBC72 Router (Title and Server)	10/11/2023	103.5.132.3	80	Router
2023-10-11 17:58:19.385654 | 45.65.196.134:80 | Title is welcome and server is None (NOID)	10/11/2023	45.65.196.134	80	Server
2023-10-11 18:13:02.806174 | 193.219.190.189:80 | Title is VDU bendrabuÃ„Â¨iÃ…Â³ interneto registracija and server is Apache/2.4.10 (Debian) OpenSSL/1.0.2l (NOID)	10/11/2023	193.219.190.189	80	Server
2023-10-11 18:27:36.544297 | 77.46.171.174:80 | MikroTik RouterOS version RouterOS v6.49.6(Login Page Title)	10/11/2023	77.46.171.174	80	Router
2023-10-11 18:30:35.169714 | 80.90.84.219:80 | Hikvision-Based DVR (Server Only)	10/11/2023	80.90.84.219	80	DVR

Table 3: Example of data in devices database

5 Description of Files

1. (Optional) Config.json: This file takes in the keys from ipinfo.io account that you create so it can extract geolocation information
2. (Optional) full_geolocations_db: geolocations database generated by optional code. It is generated new first time the code is run, but can be appended to in subsequent runs.
3. (Optional) full_devices_db: device types database generated by optional code. It is generated new first time the code is run, but can be appended to in subsequent runs

4. (Optional) ExtractUniqueSourceAddresses.ipynb: This file takes the meanie data and generates list of unique source addresses that will be queried for geolocation and/or devices. If the code is not being run for first time and there is already a full_geolocation_db and full_devices_db, then it compares the two databases to unique source addresses and generates lists of only the source IP addresses that we don't have geolocation and devices database for.
5. (Optional) GetIPGeoLocation.ipynb: This code uses ipinfo.io to extract geolocations for unique sources addresses file generated in the previous step for geolocations. It can also merge the information extracted with previous geolocations database (full_geolocations_db).
6. (Optional) ProcessDevices.ipynb: This takes output file(s) from lift and process them. It can also merge them with a previous device types database (full_devices_db).
7. GreenwichMeanieYearlyExperiments.ipynb: This runs yearly analysis on Greenwich meanie data. If a geolocations database and device types database is provided, it can also analyze that data.
8. GreenwichMeanieWeeklyDailyExperiments.ipynb: This runs weekly and daily analysis on one week of data per year. It can also run geolocation weekly analysis if that database is provided.

6 Running the Code

All the code files have a section near the top called ***'User Defined Variables'*** where the user should specify inputs/outputs and preferences. The description of each variable is listed with that variable. Note that the output folders that users specify in the code have to exist beforehand since the code will not create those folders.

The code runs in two steps:

1. Running optional code. If the user chooses to run the optional code, it has to be run before running the meanie analysis.
2. Running the Greenwich Meanie Analysis

6.1 Running the Optional Code with External Modules

The goal is to create or update the geolocations database and devices database. ***Running this code is optional*** and the following order should be followed when running this code.

1. **Run ExtractUniqueSourceAddresses.ipynb -** This takes the following inputs and output files:
   1.1. Input 1: Existing geolocations database (full_geolocations_db.csv), if one exists and being merged
   1.2. Input 2: Existing device types database (full_devices_db.csv), if one exists and being merged
   1.3. Input 3: Input_dir_path, a folder that has the meanie text files that the user wants to analyze
   1.4. Output 1: IPsForGeoLocation.txt file in output directory <output_dir_path> with all unique source IPs, or IPs that current geolocations database does not have geolocations for, if it is being merged
   1.5. Output 2: IPsForDeviceTypes.txt file in output directory <output_dir_path> with all unique source IPs, or IPs current devices database does not have device info for, if it is being merged
2. **Run GetIPGeoLocation.ipynb -** This takes the following inputs and output files:
   2.1. Input 1: Config.json. This file has ipinfo_keys parameter, a dictionary where the keys are IPinfo keys and the values are how many IP queries are left. Example: "ipinfo_keys": {"key1": 50000, "key2": 28000}.
   2.2. Input 2: geo_locations_ip_file, file name and full path to the output file described in bullet 1.4

2.3. Input 3: existing_geolocations_db, set this value to 1 in there is a previous geolocations database that is being merged and then specify the file + full path to that database file in the next variable. Set to 0 if there is no prior geolocations file.
2.4. Input 4: geo_location_db_file, file name and full path to the existing geolocations database file also used in bullet 1.1.
2.5. Output 1: Note this overwrites the existing geolocations database file described in bullet 2.4. It takes the previous geolocations, combines with geolocations of the new IP addresses, and creates a larger database of addresses.

3. **Run lift** as described by the documentation at (https://github.com/trylinux/lift). An example query: `lift -f IPsForDeviceTypes.txt -p 80 -p 443 -c 10 -o outputfile.txt`. The inputs and outputs for lift code are as follows:
   3.1. Input 1: IPsForDeviceTypes.txt generated in bullet 1.5
   3.2. Output 1: outputfile.txt (or text file with other appropriate file name defined by user) containing the information for IP addresses that device results were returned for.
4. **Run ProcessDevices.ipynb -** This takes the following inputs and output files:
   4.1. Input 1: dir_path, Full path of the directory where the user stored the output file generated in 3.2. Please note that it will read all the text files in that directory in case the user ran lift multiple times (e.g. with different ports) and has multiple output files
   4.2. Input 2: ip_file, file name and full path to IPs that were sent as input to lift, described in 1.5
   4.3. Input 3: existing_device_types_db, set this value to 1 in there is a previous device types database that is being merged and then specify the file + full path to that database file in the next variable. Set to 0 if there is no prior database file.
   4.4. Input 4: devices_db_file, file name and full path to the existing devices database file also used in bullet 1.2.
   4.5. Output 1: Note this overwrites the existing device types database file described in section 4.4. It takes the previous devices, combines with devices of the new IP addresses, and creates a larger database of addresses.
   4.6. Output 2: word_counts_file, file and full path to a csv file that will contain all the keywords and their frequencies that were in the files from 4.1. This will allow the user to see if they have any new device types they want to add to the input called device_types.

6.2 Running the Greenwich Meanie Analysis

5. **Run GreenwichMeanieYearlyExperiments.ipynb -** This takes the following inputs and output files:
   5.1. Input 1: file name and full path to the existing geolocations database file also used in bullet 1.1. Running the geolocations data analysis is optional and the user can skip this by setting analyze_geolocation to 0.
   5.2. Input 2: file name and full path to the existing devices database file also used in bullet 1.2. Running the devices data analysis is optional and the user can skip this by setting analyze_device_types to 0.
   5.3. Input 3: input_dir_path, full file path to the folder that contains all the meanie files to be analyzed
   5.4. Output 1: CityCountsByYear.csv located in output_dir_path specified by the user, contains counts of unique source addresses for each city (by unique latitude/longitude). This file is only printed if the geolocation analysis is enabled.
   5.5. Output 2: Top10Countries.xlsx located in output_dir_path specified by the user, contains unique source addresses for top 10 countries for each year specified in years_of_interest.

Includes two worksheets for actual source address counts, and percent of source addresses for that year. This file is only printed if the geolocation analysis is enabled.

5.6. Output 3: Top15Cities.xlsx located in output_dir_path specified by the user, contains unique source addresses for top 15 cities for each year specified in years_of_interest. Includes two worksheets for actual source address counts, and percent of source addresses for that year. This file is only printed if the geolocation analysis is enabled.

5.7. Output 4: YearlyCountrySources.csv located in output_dir_path specified by the user, contains all the countries with their unique source address counts for each year specified in years_of_interest. This file is only printed if the geolocation analysis is enabled.

5.8. Output 5 onwards: Graphs saved in output_dir_path specified by the user, if user chose to save the graphs by setting save_figs to 1.

6. **Run GreenwichMeanieWeeklyDailyExperiments.ipynb -** This takes the following inputs and output files:

6.1. Input 1: geo_location_db_file, file name and full path to the existing geolocations database file. Running the geolocations data analysis is optional and the user can skip this by setting analyze_geolocation to 0.

6.2. Input 2: input_dir_path, full file path to the folder that contains all the meanie text files to be analyzed.

6.3. Output 1: CityCountsByWeek.csv located in output_dir_path specified by the user, contains counts of unique source addresses for each city (by unique latitude/longitude). This file is only printed if the geolocation analysis is enabled.

6.4. Output 2 onwards: Graphs saved in output_dir_path specified by the user, if user chose to save the graphs by setting save_figs to 1.