Automate Me in Python!

Objective

Automate real-world incident response tasks.

Assignment Description

In this project you are given the role of a security analyst. You are sent botnet alerts which refers to a notice claiming that some device in your organization was detected to communicate with a known **sinkhole** [a network address used by malware]. This means that some device in your network is potentially infected with malware. Your job is to identify the actual **user** (**along with the MAC address of the device**) within the organization that has potentially been infected. Problem is that your organization has tens of thousands of users and has the additional layer of indirection through NATed routers. This situation is simply crying out to you: **AUTOMATE ME**!

Datasets

It includes the following:

1. DHCP Logs

File Format: A CSV file with the following columns.
ip_decimal, mac_string, pc_name, transaction, timestamp, primary_key

Sample data:
172.18.12.42, b6:49:d3:93:b0:9e, vwnhjk-MacBook-Pro, DHCPACK, 2016-03-10 00:00:00, 106443

Note: The MySQL database will store the IPs in decimal format. You will have to preprocess the data before you store it in the database

2. RADIUS Logs

File Format: A CSV file with the following columns.
timestamp, username, FramedIPAddress, AcctStatusType, CallingStationId
Sample data:
2016-03-10 00:00:00, hndmfwi, 172.19.153.130, Start, 80:c4:89:1e:c8:ee

3. Contact Info

File Format: A CSV file with the following columns.
mac_string, contact
Sample data:
5f:22:d4:c5:64:78, wkotfrf

4. NAT Logs

[UPDATED] **File Format:** Extract natlogs.tar. A folder containing multiple gzipped CSV files with the following columns.
Timestamp, -, pre_nat_ip, pre_nat_port, remote_ip, remote_port, post_nat_ip, post_nat_port, -, -, -, -, -
Sample data:
2016-03-20T23:00:01.671-04:00, 1804, 172.19.151.145, 60429, 152.163.0.98, 143, 192.168.226.151, 43794, 6, 476, 27112, 790, 1138539

File naming convention:
Logs for each hour are stored in a separate file. i.e. 24 log files will be generated per day. Each file is named as follows:
nat.csv.<YEAR><MM><DD><HH>.csv.gz
Sample File Name:
nat.csv.2016032100.csv.gz … nat.csv.2016032123.csv.gz

Note: Logs from 00:00:01 am will be stored in the file for the 1st hour, i.e., nat.csv.2016032101.csv.gz
Similarly, logs for 10:54 am will be in the file for the 11th hour i.e. nat.csv.2016032111.csv.gz

Manual Walkthrough

Here, we manually go through the steps taken while processing an alert.
The goal of analyzing botnet alert is to identify the device (MAC address) and the actual user. With this information, we can notify the user.

1. Terminology

Pre-NAT IP/Port: Will refer to the internal non-routable private IP/port used within the organization.
Post-NAT IP/Port: Will refer to the external public facing IP after the NAT translation.
Remote IP/Port: Will refer to the destination IP/port of the request.
Timestamp: The date and time when a certain event is logged.

2. Parsing the alert notice

From the given alert (**file**: notice1.txt) you can infer the following information:
Post-NAT IP: 192.168.226.52
Post-NAT port: 33631
Remote-IP: 195.22.28.196
Remote port: 80
Timestamp (in Zulu or UTC format): 2016-03-21T14:54:27Z

Note 1:
a. Refer to the Logs and Log Analysis slides for the explanation of the overall network.
b. The logs record timestamps in local time. In our case, all logs are recorded in Eastern time, which is 4 hours behind Zulu time.
c. **Useful tools**: regular expressions, xml parser

Timestamp (in Eastern Time): 2016-03-21T10:54:27Z

3. Identify pre-NAT IP/Port

Use the NAT logs to identify the pre-NAT IP and port associated with the above data.
Step 1: Identify the log file(s) to look at based on the local timestamp. Here, the logs for 10:54 AM will be in the file for 11th hour i.e. nat.csv.2016032111.csv.gz
Step 2: Filter the log entries.

```
[Practical Cyber Security] > zgrep "2016-03-21T10:54:" nat_logs/nat.csv.2016032111.csv.gz | grep "192.168.226.52,33631"
2016-03-21T10:54:27.572-04:00,2,172.19.52.38,65149,195.22.28.196,80,192.168.226.52,33631,6,6,1062,6,595
```

The command basically filters the log entries first by time, then by source IP and source port.

Note 2:
a. These logs are stored in compressed form to save disk space. You can use tools that work directly with compressed files.
b. **Useful tools**: The example above uses zgrep. It invokes grep on compressed or gzipped files.
http://www-01.ibm.com/support/docview.wss?uid=swg21996814
https://linux.die.net/man/1/zgrep
c. The notice provided might not have accurate timing information. You will have to setup a search window of +/- 10 mins.
- If no entry is found within the window what should you do?
- If multiple entries are found what should you do?

We now have the following new information.
Pre-NAT IP: 172.19.52.38

4. Identify the MAC address

We can now go to DHCP logs to identify the MAC address of the device.

[UPDATED NOTE]: We have already converted the ip in the DHCP logs to the decimal representation. So, the decimal representation of IP: 172.19.52.38 is equal to 2886939686. Hence, we grep for the decimal representation here.

```
[Practical Cyber Security] > grep "2016-03-21 10:" dhcp_logs.csv | grep "2886939686"
2886939686,d3:1e:c8:b4:59:47,voxpeo-Blackberry,DHCPACK,2016-03-21 10:53:37,199655
2886939686,d3:1e:c8:b4:59:47,voxpeo-Blackberry,DHCPACK,2016-03-21 10:53:57,106661
2886939686,d3:1e:c8:b4:59:47,voxpeo-Blackberry,DHCPACK,2016-03-21 10:54:17,199655
```

We now have the MAC address of the potentially infected device.
MAC address: d3:1e:c8:b4:59:47

Note 3:
a. The DHCP logs will be stored in MySQL database (see Assignment Setup section for more information).
b. **Useful tools**: MySQL connector

5. Identify the User

The last information we need is the username.

Note 4:
User information if stored in two different tables in MySQL database based on which subnet the user connected to.
a. RADIUS logs: for subnet 172.19.*.*
b. Contact info: for others

Here, we use the RADIUS log as the pre-NAT IP is in subnet 172.19.*.*.

```
[Practical Cyber Security] > grep "2016-03-21 10:" radius_logs.csv | grep "172.19.52.38"
2016-03-21 10:53:55,hzmavog,172.19.52.38,Stop,d3:1e:c8:b4:59:47
2016-03-21 10:53:57,hzmavog,172.19.52.38,Start,d3:1e:c8:b4:59:47
```

Do a sanity check by comparing the MAC address from DHCP logs with RADIUS logs. In this case they seem to be matching and hence we have the right user.
Username: hzmavog

Assignment Setup

Setting up MySQL

You will need to install MySQL database. A script is provided to create the schema for each of the logs that will be stored in this database, namely DHCP, RADIUS and Contact Info.
Make sure that your MySQL database is running and execute the script provided as follows.

```
$ ./create_table.sh
```

Check the database and verify that the tables have been created.
Another script is provided to load the csv files into tables.

```
$ ./insertcsv.sh
```

Note: You will have to edit this script to fill in the path to the csv file you want to insert.

Programming Environment

You are **only allowed to use Python**. Do keep in mind that you will be required to do the following in this assignment:

1. **Connect to a MySQL database.**
2. **Parse XML files.**
3. **Time zone related manipulations.**

You must provide a detailed **README** file to describe what libraries you use and how to use your script.

Assignment Requirements

Implementation Details

The notices will be provided as text files. Your program should:

1. Take the path to the directory containing these files as input from the command line.
2. Read, parse and process each of these notices automatically.
3. Output the results to the console/text file in an easy to digest format.

E.g .,

```
$ python automator.py <path/to/directory/with/all/the/notices>
```

You may assume that the nat logs are stored in a folder called nat_logs in the directory with the script. Sample directory structure for the project.

```
.
|-- automator.py
|-- readme.txt
|-- project_report.pdf
|-- nat_logs
    |-- nat.csv.2016032100.csv.gz
    |-- nat.csv.2016032101.csv.gz
```

Coding Style

The premise of this assignment is that you are working in a real SOC. So, the script you develop must be easily understood and used by others as well. Points will be allocated for clarity in code structure, modularity and proper comments.

Readme

As mentioned before, a proper readme must be provided which consists of at least the following information:

- Dependencies: List all the libraries used and installation guideline (if any).
- Running the code: Instructions on how to run the code, including compilation instructions (step by step process)
- Any other information we may need to know. For example, any additional assumptions you may have made while development.

Report

A concise report documenting your thought process through this assignment. Also, include screenshots showcasing the results or findings when running the script.