Table of Contents

Package Details

1. convert.py: Handles the offline processing of log file and saving the results to both text and csv files.
2. IHL.py: Encapsulates the IHL class-specific methods for uniqueUserFiles I/O and specific variables for keeping track of usage statistics during the log file extraction.
3. convertSingaren.py: Extracts data concerning number of individual authentication requests from the logfile and save to a single csv file.
4. CreateHTML.py: Writing of html files for each IHL.
5. CreateHTML_singaren.py: Writing of html file for singaren staff to check eduroam server.
6. ihlconfig.json: Configuration file containing IHL's server names and IP addresses. Includes Euro Top-level servers too.
7. Daily/Monthly/Yearly CSV files in csv folder contain the data source for the visualisation. Need to be in same directory as the html files.
8. ServerLoad CSV file contain the data source for the visualisation in the overall stats page for singAren staff. (total.html)

Detailed Description of Modules

Convert.py

Inputs: Radsecproxy log file, UniqueUsers Files for all IHLs, ihlconfig.json config file

Outputs: UniqueUsers Files for all IHLs, Stats results log file, Daily/Monthly/ Yearly csv files

Dependencies: ihlconfig.json, IHL.py

Functions:

- **logExtract (logData, IHL_Array, etlr_server, etlr_ip)**: Defines the logic in extracting information from the log file. The log data is sorted into accepted and rejected status for each entry and for each IHL, the number of visitors and local users are collated.
- **results (IHL_Array, filename)**: Writes the results in a daily log file with date attached to it. The results file contains for each IHL, the number of localUsers overseas and at other IHLs, the number of visitors accessing eduroam at that IHL, the total number of accepted/rejected unique users for the current month and year.
- **saveCSV (IHL_Array, filename, interval, previous_date)**: Saves the extracted data into daily, monthly and yearly CSV files for data visualisation.
- **Is_non_zero_file (fpath)**: A helper function used by saveCSV to check for empty or non-existent files.
- **Main()**: Main function controlling the program flow and coordinating the operation of all the different functions.

NOTE: Check the filepath input before running to ensure that the radsecproxy file is in the right working directory.

i.e. log_file = open(“radsecproxy.log_”+day+month+year_2numbers,”r”)

ConvertSingaren.py

Inputs: Radsecproxy log file

Outputs: ServerLoad csv file

Functions:

- Class **ServerLoad**:
    - Constructor __init__: Initialises accepted and rejected lists of 24 elements, one for each hour.
    - Functions:
        - **update(self,array,time)**: Updates the list elements for every hour of the log to collect the number of requests for that time period
        - **saveCSV(self,filename,date)**: Save the extracted data into a CSV file logging number of requests hourly
        - **logExtract(self,logData)**: Defines the logic in extracting the information of the number of accepted/rejected requests from the log file
- **Main():** Main function controlling the program flow and coordinating the operation of all the different functions.

NOTE: Check the filepath input before running to ensure that the radsecproxy file is in the right working directory.

i.e. log_file = open("radsecproxy.log_"+day+month+year_2numbers,"r")

IHL.py

Constructor __init__:

- Inputs/Outputs:
  - UniqueUsers<IHL>.log_MMYY, rejectUniqueUsers<IHL>.log_MMYY, UniqueUsers<IHL>.log_YY, rejectUniqueUsers<IHL>.log_YY
- Arguments: name, ipAddress, server, filepath (folder where uniqueUserFiles are found)
- Variables:
  - userRecordsMonth: Set of unique users for the month
  - userRecordsYear: Set of unique users for the year
  - rejectRecordsMonth: Set of reject unique users for the month
  - rejectRecordsYear: Set of reject unique users for the year
  - reject_localUsers: Number of rejected local users for the day
  - reject_visitors: Number of rejected visitors for the day
  - localUsers: Number of accepted local users for the day
  - visitors: Number of accepted visitors for the day.
  - localUsersCount: Dictionary variable containing number of users at each other ihl and overseas.
  - Localvisitors: Number of accepted visitors from other local ihls for the day.

Functions:

- **readUniqueUserFiles(self, month, year**): Open all the unique user files associated with the IHL (accepted, rejected, monthly, yearly) and save the contents into the respective IHL variables. Uses readFile() function.
- **readFile(self, filename):** Read a single unique user file and save in a list named "records". Returns a list "records".
- **writeUniqueUserFiles(self, month, year):** Write the unique users back to the unique user files (accepted, rejected, monthly, yearly). Uses writeFile() function.
- **writeFile(self, filename, userRecords):** Write the list "userRecords" into a single unique user file. Returns a file.
- **getUniqueCountMonth(self):** Get number of unique users from the IHL for the month
- **getUniqueCountYear(self):** Get number of unique users from the IHL for the year
- **getRejectUniqueCountMonth(self):** Get number of reject unique users from the IHL for the month.
- **getRejectUniqueCountYear(self):** Get number of reject unique users from the IHL for the year.

- **getRejectCount(self):** Get total number of rejected user from the IHL for the day.
- **getLocalVisitors(self):** Get the number of visitors from local IHLs accessing eduroam at this IHL alone.

CreateHTML.py

Inputs: DailyMMMYYYY.csv, MonthlyYYYY.csv, Yearly.csv (needed by the html files)

Outputs: 1 HTML file for each IHL

Dependencies: D3 library=> d3.min.js, DimpleJS library=> dimple.v2.1.3.min.js, ihlconfig.json(for initialising ihls)

Functions:

- createHTML (ihlName, day, month, year): Creates the html template code of the usage stats for IHL with data visualisation.
    - Header: Starting html code of the webpage. Contains definitions for localUsers, visitors and rejected. Also contains script dependencies i.e.

      ```
      <script src=" https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.5/d3.min.js"></script>
      ```

      ```
      <script src="http://dimplejs.org/dist/dimple.v2.1.3.min.js"></script>
      ```

    - dailychart: Graph for daily data for the current month. Uses Daily CSV file.
    - monthlychart: Graph for monthly data for the current year. Uses Monthly CSV file.
    - yearlychart: Graph for yearly data. Uses Yearly CSV file.

Definitions:

- Local Users: IHL A staff/students using eduroam outside of IHL A.
- Visitors: Users from other IHLs and Foreign exchange users who are using eduroam in IHL A.
- Rejected: Includes all rejected local users and visitors in IHL A.
- Statistics do not include local users using eduroam in their own facilities.

CreateHTML_singaren.py

Inputs: ServerLoadYYYY.csv

Outputs: 1 HTML file for singAren staff to check eduroam server load. (total.html)

Functions:

- createHTML (ihlName, day, month, year): Creates the html template code of the usage stats for IHL with data visualisation.
    - Header: Starting html code of the webpage. Contains script dependencies i.e.

      ```
      <script src=" d3.min.js"></script>

      <script src=" dimple.v2.1.3.min.js"></script>
      ```
    - Hourlychart: Graph for hourly data for the day.
    - dailychart: Graph for daily data for the current month.
    - monthlychart: Graph for monthly data for the current year.
    - All 3 charts depend on ServerLoad CSV file for data source.

CSV Files

Filename: Daily<Month><Year>.csv

    Date, IHL, Users, Category(LocalUsers,Visitors,Rejected)

Filename: Monthly<Year>.csv

    Month, IHL, UniqueUsers, Category(Accepted,Rejected)

Filename: Yearly.csv

    Year, IHL, UniqueUsers, Category(Accepted,Rejected)

Filename: ServerLoad<Year>.csv

    Date, Month, Hour, Requests, Category(Accepted,Rejected)

Installation Guide

1. Download and install the latest Python distribution, i.e. Python version 3.4.3 at http://www.python.org/downloads/
2. Download the DimpleJS library from www.dimplejs.org . Locate dimple.v2.1.3.min.js under the dimple/dist folder.
3. Download the D3.js library from www.d3js.org. Locate d3.min.js in the folder.

Operation Guide

1. Copy log file from Eduroam FLR (IP: 203.30.39.51) and rename the copy with the previous day date at 0000hrs.
2. Transfer the copied file to Eduroam Stats (IP: 203.30.39.58) for processing.
3. Run convert.py to extract information from the log file and store the results into Daily/Monthly/Yearly CSV files for data visualisation.
4. Run convertSingaren.py to extract information from log file and store the total stats into ServerLoad CSV file.
5. Run CreateHTML.py to create the html files with the graphs for each IHL. Check that all the IHLs are represented in the ihlNames list in the python file. Also check the filepath of the csv files in the html code is correct.
    a. Check this part eg: d3.csv("Daily"""+month+year+""".csv", function (data)... for dailychart, monthlychart and yearlychart.
6. Run CreateHTML_singaren.py to create the html file for the eduroam server load stats for singaren staff.
7. Transfer the html files on to the Web server and the csv files in the same directory as the html files via FTP. (IP: 203.30.39.3).
8. Web Server displays the information on webpages of the singaren website.
   (e.g. http://www.singaren.net.sg/stats/nus.html)

Maintenance Tips

1. Before running convert.py, please check the filepath input before running to ensure that the radsecproxy file is in the right working directory. The same goes for the other files that need to be accessed by convert.py.

i.e. log_file = open(“radsecproxy.log_”+day+month+year_2numbers,”r”)

results(IHL_Array,"Stats_results/results.log_"+day+month+year_2numbers)

saveCSV(IHL_Array,'csv/Daily'+month_words+year,'Day',previous_date)

saveCSV(IHL_Array,'csv/Monthly'+year,'Month',previous_date)

saveCSV(IHL_Array,'csv/Yearly','Year',previous_date)

2. In order for the HTML graphs to work on localhost, which means no internet access, change the script dependencies to the dimple and d3 libraries in the local directory:

Eg: Change

<script src="https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.5/d3.min.js"></script>

<script src="http://dimplejs.org/dist/dimple.v2.1.3.min.js"></script>

To:

<script src=" d3.min.js"></script>

<script src=" dimple.v2.1.3.min.js"></script>

3. To include a new IHL into the log processing system:

- Save the ip addresses and server name of the IHL in ihlconfig.json file in this format:
  - "ihl":{ "ip": [*list of ip addresses*],

    "server": [*list of server names eg IHL_IdP_SP*] }
- This will automatically include the new IHL into the convert.py and the CreateHTML.py program to generate the eduroam statistics.

Process Flow of Statistics Program

Convert.py Process (Part 1)

1. Open the copied log file "radsecproxy.log_*previous day date*"
2. Save all the information from log file into a variable logData, a list of log entries in the file and close the file.
3. Load the ihlconfig.json file into the program. The json file contains the IP addresses and Server names for all the IHLs and the Euro TLR servers.
4. Initialise all the IHLs into a dictionary IHL_Array to facilitate the log extraction process.
5. Load uniqueUsers.log from subdirectory "uniqueUsersFiles" for each month, year and each IHL (done through IHL.py method readUniqueUserFiles)
    a. Os.path.isfile checks whether the specific uniqueUsers.log is a regular file and not a directory.
        i. If true, open the file for reading and create uniqueRecords as a set of existing unique users to check with the extracted log file.
        ii. If false, open the file for writing the new set of unique users. uniqueRecords will be an empty set.
    b. Close each uniqueUsers.log file
6. Run Log Extract.For each entry in logData, use regex to search for 'Access-Accept/Reject for user'
    a. re.search (r'Access-Accept for user', line, re.MULTILINE|re.IGNORECASE)
    b. If Access-Reject, matchReject is set.
        i. If user not in daily_RejectedRecords, add user to it.
        ii. Check where user is coming from (Identity Provider) and where he is going to (Service Provider/Access Point)
        iii. Keep count of rejected users, whether local or overseas, for each IHL.
    c. If Access-Accept, matchAccept is set
        i. If user not in daily_AcceptRecords, add user to it.
        ii. Check where user is coming from (Identity Provider) and where he is going to (Service Provider/Access Point)
        iii. Keep count of localusers overseas, in different IHLs, for each IHL they come from through the variable IHL_Array[ihl].localUserCount.
        iv. Keep count of overseas visitors accessing the network for each IHL.
7. Total the number of rejected localUsers and visitors from each IHL in IHL_Array[ihl].reject_localUsers and IHL_Array[ihl].reject_visitors.
8. Total the number of localusers in different IHLs from each IHL in IHL_Array[ihl].localUsersCount.
9. Open the uniqueUsers.log file for each month, year, each IHL, reject/accept.
    a. For each user in the uniqueRecords, count the number of unique users for the IHL.
    b. Write back the unique user names in uniqueUsersFile.
10. Create a log file "results.log_*date*" to store the total number of localUsers from each IHL and the total number of visitors to each IHL.
11. Results content should contain

   a. Total number of localUsers from NTU/NUS/SMU/ASTAR/NIE who are abroad and in other IHLs. Also total number of localUsers from each IHL in total.
   b. Total number of visitors to NTU/NUS/SMU/ASTAR/NIE.
   c. Total number of unique/rejectUnique users to each IHL for the month.
   d. Total number of unique/rejectUnique users to each IHL for the year.
   e. Total number of rejected accesses from each IHL for the day.

Convert.py Process (Part 2: Saving to Daily,Monthly,Yearly CSV files)

1. Open the csv file to extract the present list of usage statistics into the local list variable “csv_list”.
2. If the csv file is not found, create a new csv file and close it. Add in the first row of labels in the list “csv_list”
   a. If file is daily, add [“Date”,”IHL”,”Users”,”Category”] as labels. Categories are LocalUsers, Visitors and Rejected.
   b. If file is monthly, add [“Month”,”IHL”,”UniqueUsers”,”Category”] as labels. Categories are Accepted and Rejected.
   c. If file is yearly, add [“Year”,”IHL”,”UniqueUsers”,”Category”] as labels. Categories are Accepted and Rejected.
3. Check for duplicate entries before appending the stats to the list “csv_list” by using the first item of the last row as reference. Set it as the variable “last_checked”.
   a. For daily file, check if the current date is the same as “last_checked” and “last_checked” is not the label “Date”. If true, we filter out the duplicate data and append the new data to the list.
   b. For monthly file, check if the current month is the same as “last_checked” and “last_checked” is not the label “Month”. If true, we filter out the duplicate data and append the new data to the list.
   c. For yearly file, check if the current year is the same as “last_checked” and “last_checked” is not the label “Year”. If true, we filter out the duplicate data and append the new data to the list.
4. After the duplicate check, we can append the new statistics values to the list “csv_list”
   a. For daily file, append the numbers of localUsers, Visitors and Rejected Users as each row for each IHL to csv_list.
   b. For monthly file, append the numbers of Accepted UniqueUsers for the month and the numbers of Rejected UniqueUsers for the month as rows, for each IHL to csv_list.
   c. For yearly file, append the numbers of Accepted UniqueUsers for the year and the numbers of Rejected UniqueUsers for the year as rows, for each IHL to csv_list.
5. Write back the contents of csv_list to the csv files. (File format is DailyMay2015.csv, Monthly2015.csv and Yearly.csv).

CreateHTML.py Process (Data Visualisation on the singaren stats webpages for each ihl)

1. Set the previous date and the filepath to ensure the right csv files are referenced by the data visualisation tool. (filepath='html files/' by default)
2. Get the names of the IHLs involved from ihlconfig.json and store them in the variable ihlNames.
3. Open the html files for each IHL and overwrite those files with the new HTML code produced by the createHTML() function.
4. In the createHTML() function run for each IHL,
    a. Insert source links to external libraries used for drawing the graphs
        i. https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.5/d3.min.js for D3 dependency
        ii. http://dimplejs.org/dist/dimple.v2.1.3.min.js for dimple library
    b. Create daily graph for the current month with the combination of Line plot and Scatter plot to show a line graph with clear indicated values. Data-source is obtained from the Daily CSV file.
    c. Create monthly graph for the current year with a stacked bar plot graph showing Accepted and Rejected UniqueUsers. Data-source is obtained from Monthly CSV file.
    d. Create yearly graph for the IHL with a stacked bar plot graph showing Accepted and Rejected UniqueUsers. Data-source is obtained from Yearly CSV file.
    e. Return the whole HTML code as a string.
5. The process of graph creation is described in the following sequence:
    a. Create new SVG element of size 800 x 600px with dimple for the chart area.
    b. Get the CSV file on the server via HTTP GET request. (done by D3 API)
    c. Filter Data according to the IHL name under the column "IHL".
    d. Initialize the x-axis and y-axis using addMeasureAxis/addTimeAxis/addCategoryAxis.
    e. If TimeAxis is used for daily chart, set the minimum and maximum values with overrideMin/overrideMax and set the time period to "d3.time.days".
    f. Set tickFormat of y-axis to ensure the absolute values of the data.
    g. Add the CSV data to the chart with addSeries and dimple.plot decides the type of graph.
    h. Add the legend for the graph with addLegend and use mychart.draw to create the full graph.
6. Write and save the HTML code string to the html files for all the IHLs.

convertSingaren.py

1. Open the copied log file “radsecproxy.log_*previous day date*”
2. Save all the information from log file into a variable logData, a list of log entries in the file.and close the file.
3. Initialise the class serverLoad with its class variables, accepts and rejects, to keep track of the number of accepted/rejected requests hourly.
4. Run Log Extract.For each entry in logData, use regex to search for ‘Access-Accept/Reject for user’
    a. re.search (r’Access-Accept for user’, line, re.MULTILINE|re.IGNORECASE)
    b. If Access-Reject, matchReject is set.
        i. The variable rejects is incremented for the specific hour.
    c. If Access-Accept, matchAccept is set
        i. The variable accepts is incremented for the specific hour.
5. Open the csv file to extract the present list of usage statistics into the local list variable “csvlist”.
6. If the csv file is not found, create a new csv file and close it. Add in the first row of labels in the list “csvlist”, ["Date","Month","Hour","Requests","Category"].
    a. Check for duplicate entries before appending the stats to the list “csvlist” by using the first item of the last row as reference. Set it as the variable “last_checked”. check if the current date is the same as “last_checked” and “last_checked” is not the label “Date”. If true, we filter out the duplicate data and append the new data to the list.
7. After the duplicate check, we can append the new statistics values to the list “csvlist”
    a. Append the number of requests for each hour of the day as each row to csvlist.
8. Write back the contents of csvlist to the csv files. (File format is ServerLoad<year>.csv).

CreateHTML_singaren.py Process (Data Visualisation on the singaren stats webpages for total.html)

1. Set the previous date and the filepath to ensure the right csv files are referenced by the data visualisation tool. (filepath='html files/' by default)
2. Open the total.html file and overwrite it with the new HTML code produced by the createSingarenHTML() function.
3. In the createSingarenHTML() function run for each IHL,
    a. Insert source links to external libraries used for drawing the graphs
        i. https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.5/d3.min.js for D3 dependency
        ii. http://dimplejs.org/dist/dimple.v2.1.3.min.js for dimple library
    b. Create hourly graph for the current month with the area plot to show a line graph with clear indicated values. Data-source is obtained from the ServerLoad CSV file.
    c. Create daily graph for the current year with the area plot graph showing Accepted and Rejected requests. Data-source is obtained from ServerLoad CSV file.
    d. Create monthly graph for the IHL with a stacked bar plot graph showing Accepted and Rejected requests. Data-source is obtained from ServerLoad CSV file.
    e. Return the whole HTML code as a string.
4. Write and save the HTML code string to the html files for total.html.