Origin_Server

Release 2.1

CABOS Matthieu

Jan 28, 2022

CONTENTS

Author *CABOS Matthieu*

Date *2021/2022*

Organization *ICGM-CNRS*

These Scripts have been written to manage properly an Origin Server (See Origin)

It is adapted to the Origin ssh platform

(reading and treating **opt/Linux_FLEXnet_Server_ver_11.16.5.1/Licenses** and **/usr/local/flexlm/orglabdebug.log** wich are the Licence File and the Tokens Log file)

These scripts need a ssh session access into the origin server (with form origin.domain.fr)

These main scripts have been written to automate the DHCP Informations retrievment and Origin Server essentials informations.

The main project is made of the following files :

- **Get_Origin_Info**
- **Origin_API**
- **Treat_tokens**
- **Treat_log_v2**

The others file concern two pre-versions of the project. Each of them is associated to its API:

- **Origin_Users_parallelisation_v2**
- **Origin_Users_parallelisation_v2 associated API**
- **Get_User_Info_From_IP_v2**
- **Get_User_Info_From_IP_v2 associated API**

CHAPTER

ONE

GET_ORIGIN_INFO_V2.1.SH

```
Get_Origin_Info_v2.1.sh
```

This is the finale version of the project. It give us the same informations as the Get_Origin_Info python files in less than 1 second. It use a subshell by connected user to treat a large amount of users without a loss of time.

The algorithm used is similar as the python files, in fact, the optimisation is ruled by the tftp request ssh remote command (line 28). The same output will be used to get almost all the needed informations as :

- **Username**
- **Ip address**
- **Cisco Switch Name**
- **Vlan Identifier**
- **Mac address**
- **Socket Identifier (with form Gix/y/z)**
- **Connexion_time (in minuts)**

The last information (description of the outlet, with form N1A01-01) is extracted from a last cisco ssh remote command defined line 43. The differents informations are extracted by a regular expression filtering defined by :

- **Username** : *[A-Za-z0-9_-êïù]+@[A-Za-z0-9_-]+*
- **Ip address** : *([0-9]+\.){3}[0-9]+*
- **Cisco Switch Name** : *balard-[0-9][A-Z]\-[0-9]*
- **Vlan Identifier** : *\s[0-9]{3}\s*
- **Mac address** : *([0-9a-f]{4}\.){2}[0-9a-f]{4}*
- **Socket Identifier** : *Gi([0-9]\/){2}[0-9]+*
- **Connexion_time** : by the following equation *M1-M2 + (H1-H2)*60* where :
  - **H1** : *The Now date command extracted time Hours field*
  - **H2** : *The Start Connexion Time Hours field*
  - **M1** : *The Now date command extracted time Minuts field*
  - **M2** : *The Start Connexion Time Minuts field*
- **Description** : *[NRJPASEP]+[0-9A-Z.]+\-[0-9]+*

The main algorithm is ruled by the following steps :

- **Get the user:ip association** from the results of the Treat_log_v2.1.sh logfile analizer script

- **Get the ip adress** : The ip address is extracted from the Treat_log_v2.1.sh script by applying the corresponding regular expression to its output
- **Use the ip adress to request the tftp server** : Send a request to the tftp server as a ssh session with remote command as argument : grep $ip /var/lib/tftpboot/snoop/where $ip contains the previous result
- **Filter the cisco name by regular expression pattern** : The Cisco name is extracted from the tftp response by applying the corresponding regular expression
- **Filter the Vlan identifier by regular expression pattern** : The Vlan identifier is extracted from the tftp response by applying the corresponding regular expression
- **Filter the MAC adress by regular expression pattern** : The MAC address is extracted from the tftp response by applying the corresponding regular expression
- **Filter the Socket name by regular expression pattern** : The Socket name is extracted from the tftp response by applying the corresponding regular expression
- **Get origin name** : From the splitted current item (separator is ':')
- **Get the time informations associated to the user** : Get the corresponding line in the time_list by filteering with previous name result
- **Get the time "now"** : Get the absolute 'now' time from the command **`date +%H`":"`date +%M`**
- **Get the "now" hours field** : Split the First field of the Now time from separator **':'**
- **Get the "start" hours field** : Split the First field of the Start time from separator **':'**
- **Get the "now" minuts field** : Split the Second field of the Now time from separator **':'**
- **Get the "start" minuts field** : Split the Second field of the Start time from separator **':'**
- **Check if user is connected** : Brownse the response of the origin ssh remote command : **/opt/Linux_FLEXnet_Server_ver_11.16.5.1/lmutil lmstat -a -c /opt/Linux_FLEXnet_Server_ver_11.16.5.1/Licenses/Origin_20jetons.lic | grep "^.*origin.srv-prive.icgm.fr/27000.*"**
  - **Use cisco and socket fields to connect the switch and get description field** : From the command **ssh ${Cisco^} "show interfaces description | i "$Socket" | tail -1"**
    - **${Cisco^}** is the Cisco name with upper first letter
    - **$Socket** is the socket number with form **Gix/y/z**
  - **Filter the Description by regular expression pattern** : Apply the Description regular expression from the previou result
  - **Compute connexion time** : From H1, H2, M1, M2 with the equation **M1-M2 + (H1-H2)*60**
- **Results display** : Sort the differents informations fields into a string
- **Put in on screen** : Put them on screen OR write it in file

CHAPTER

TWO

GET_ORIGIN_INFO.PY

2.1 Version 2.2

The update concern the used versions of Origin_API (the version 2 will be used here) and the Treat_log script (the version 2.1 will be used here). Once the newest extensions loaded, the results are treated in real time and the execution time is about 20 seconds.

To use as an history manager, please to launch directly on the remote server (the one wich can access the origin server) with the correct ssh anthorizations keys from the command :

```
nohup python3 ./Get_Origin_Info_v2.2.py
```

The results will be stored twice :

- One version of the history is stored in the local folder
- The other one is stored into the origin server root

To stop the nohup process, please to respect the path to follow :

- Identify the process PID using the command **ps -aux | grep nohup**
- Kill the process as root with the command **sudo kill -9 <PID>**

2.2 Version 2.1

The most efficient version using the Origin_API extension.

It write the orgin activities history containing all the DHCP extracted informations in concordance with the tftp boot server (listing only **real** connected users).

In fact, you have to use it into a similar structure as following :

```
Origin
├── Get_Origin_Info_v2.1.py
├── Origin_API.py
├── Treat_log_v2.sh
└── Treat_tokens.sh
```

These actions need an efficient log file since the Origin server orglabdebug.log file.

I use a logwatch intermediate file with allocated token inserted into the token log file.

It is ruled by a shell automated script containing the following instructions:

```
date >> ./logwatch
ss -n -t | grep 60213 >> ./logwatch
tail -n 1 /usr/local/flexlm/orglabdebug.log >> ./logwatch
```

where **60213** is the communication port number of the Origin application.

This shell script is lauched periodically with the following linux commands. It must be launched with the **nohup** linux command to make it write properly and permanently the logwatch file.

```
inotifywait -q -m -e modify /usr/local/flexlm/orglabdebug.log|
while read -r filename event; do
 ./Script.sh
done
```

With this way of work, the orglabdebug.log file and the logwatch file will never be altered.

This script require differents ssh authorizations keys as :

- **Cisco Switch connected to the network** *(All the Balard-XY-Z switch access)*
- **tftp.srv-prive.icgm.fr** *(All the daily connected repertored users)*
- **origin.srv-prive.icgm.fr** *(The main origin server)*

With access to these ssh passerel you will be able to retrieve all the needed informations to identify and keep tracability on your Origin server.

2.3 Version 2

That version is similar to the version 1. There is no display in this one but the Origin History file is properly written. The version 2 is treating the **full daily logwatch content** and should be used at the end of a day for exemple or to verify the results of the version 2.1.

It give us the **same informations than the version 2.1** but it will display into the Origin history file **all the activities** on the **origin server**.

It must be consider as a **pre-version of version 2.1** and should be used also as a **log file analyzer**.

Susbtitute the differents date variables (as *day, month, year, etc*) **with a specific date** will treat the logwatch file **since this specific date**.

This script will be used as a logwatch analizer instead of a real time analizer like the version 2.1 and it could be really interesting with **network management** and **administration tool**.

Please to use with the correct following syntax :

```
python3 Get_Origin_Info_v2.py
```

2.4 Version 1

This script is the full optimised and parallelized code version of the Origin Users Informations Getter. It allow us to get since an Origin server and the tftp server repertoring connected people the full informations content since the log description to the connection time.

It uses :

- **Treat_log_v2.sh** file to get an immediate association between user ID and their IP.
- **Treat_tokens.sh** script to get a tokens manager into your Python Code
- **Get_Connexion_Time.py** Library to get the connexion time elapsed by user.
- **Get_tftp_infos.py** Library to treat and manage a Tftp content from the server

It must be used into the equivalent environment :

```
.
├── dhcpd-501.conf
├── dhcpd-510.conf
├── dhcpd-511.conf
├── dhcpd-512.conf
├── dhcpd-513.conf
├── dhcpd-514.conf
├── dhcpd-515.conf
├── dhcpd-516.conf
├── dhcpd-518.conf
├── dhcpd-519.conf
├── dhcpd-524.conf
├── dhcpd-525.conf
├── dhcpd-526.conf
├── dhcpd-528.conf
├── dhcpd-529.conf
├── dhcpd-530.conf
├── dhcpd.conf
└── Origin_Manager
    ├── Get_Connexion_Time.py
    ├── Get_Origin_Info.py
    ├── Get_tftp_infos.py
    ├── Treat_log_v2.sh
    └── Treat_tokens.sh
```

This Script use the already written associated script. The ssh sessions connections have been parallelized to make the script faster than ever.

The algorithm follow these steps in order :

- **Get the logwatch file**
- **Treat the Treat_log_v2.sh output** since regular expressions to get the correct user2ip list
- **Get the Snoop dictionnary** since the tftp server of connected people (cf DHCP Snooping)
- **Get the connection time** since the *Get_Connexion_Time* library

Please to use with the correct following syntax :

```
python3 Get_Origin_Info.py
```

CHAPTER

THREE

ORIGIN_API

Author *CABOS Matthieu*

Date *2021/2022*

Organization *ICGM-CNRS*

3.1 ssh_session

```
def ssh_session(cisco,command,return_dict)
```

3.1.1 Definition

Configure and execute a SSH session with remote commands (not an option.)

It is an automatic authentified ssh session, using the environment parameters as :

- **Home absolute way**
- **user from environment variables**
- **ssh keyfile from the given absolute way**

The results will be stored into the return dict dictionnary using the Python Multithreading functions.

Parameters	Type	Description
cisco	*Str*	The name of the Cisco Switch to connect
command	*Str List*	The String command list to send to the cisco switch
return_dict	*Dictionnary*	The dictionnary storing commands output by Cisco name

3.1.2 Returns

Dictionnary

The dictionnary linking to a Cisco switch name as a key its commands list output from console.

3.2 Treat_out

```
def Treat_out(output)
```

3.2.1 Definition

Treating shell command ouptut since the tftp Boot informations reading.

To do so, this function is ruled by regular expression as :

- *regex_ip* : **([0-9]+\.){3}[0-9]+**
- *regex_mac* : **([a-zA-Z0-9]{4}\.){2}[a-zA-Z0-9]{4}**
- *regex_socket* : **Gi([0-9]+\/){2}[0-9]+**
- *regex_vlan* : **\s[0-9]{3}\s**
- *regex_switch* : **Balard-[0-9A-Z]+\-[0-9]+**

This method treat a commands list output from a ssh session with a cisco switch. It read and treat in multiline mode every met values from regular expression and store these informations into the returned dictionnary. The returned dictionnary is builded with the ip as key and following informations as values :

- **MAC address**
- **Cisco GigabitEthernet socket (with form Gix/y/z)**
- **The Vlan identifier as Integer**
- **The switch name as String**

Parameters	Type	Description
output	*Str*	The raw commands list output from the ssh session with a cisco switch

3.2.2 Returns

Dictionnary

The builded dictionnary linking to an ip as key the Cisco informations

3.3 Get_Description

```
def Get_Description(Snoop_Dict)
```

3.3.1 Definition

Get the full plug name since the Snoop dictionnary present into the tftp server (into the *var/lib/tftpboot/snoop/* repertory).

Only the real connected users will be repertoried here since the snoop tftp boot repertory. This methos has been partially coded with a parallel section to treat ssh connection faster.

This method is following this algorithm :

- **Building Cisco Instructions list** by Switch (stored into the *tmp* variable)
- **Manage the multiprocessing section** of the code with the splitted Switch Dictionnary and the shared return dictionnary to store results of ssh sessions.
- **Launching the multiprocess list** with the correct method *ssh_session* and associated builded Cisco instructions list.
- **Start and join the differents process** and rebuild the return dictionnary sorted by Cisco Switch name
- The results of **the multiples ssh session give us the full outlet description name** (with form N1A01-01) by regular expression filtering
- **Build the Description_dictionnary** linking to a Cisco gigabitEthernet socket (Gix/y/z) as key its outlet exact description.

Parameters	Type	Description
Snoop_Dict	*Dict*	The snoop dictionnary extracted from the tftp server

3.3.2 Returns

Dictionnary

The builded dictionnary associating to a cisco gigabit ethernet socket *(Gix/y/z)* its exact outlet description name as String.

3.4 cut_dic

```
def cut_dic(IPSwitchs,div)
```

3.4.1 Definition

Utilitary method to split properly and in adequation with the multiprocessing parameters the given dictionnary.

Split Dictionnary into div differents dictionnary.

Parameters	Type	Description
IPSwitchs	*Dict*	The shared dictionnary associating to a Cisco switch name its IP address.
div	*Integer*	The number of slices to build from the given dictionnary

3.4.2 Returns

List

A list of dictionnary containing the main dictionnary splitted into div differents sections.

3.5 get_Dict

```
def get_Dict()
```

3.5.1 Definition

Get the main informations dictionnary repertoring these following field:

- **IP address as key**
- **MAC address**
- **Cisco Socket**
- **Vlan Identifier**
- **Cisco Switch name**
- **Outlet Description**

The five firsts informations are extracted from the tftp boot server to get exact real values from cisco switch. The last one is extracted from Cisco switch multiple requests.

3.5.2 Returns

Dictionnary

The Snoop dictionnary repertoring all the needed network informations.

3.6 reverse

```
def reverse(line)
```

3.6.1 Definition

Perso reverse list function

Parameters	Type	Description
line	*Str List*	A string line as list

3.6.2 Returns

Str List

The reversed list

3.7 pop_double

```
def pop_double(line)
```

3.7.1 Definition

Pop double from list and build full hostname.

To do so, we have to follow these instructions :

- *Read the current line and store the doubled value as host*
- *Course te rest of list and extract the user name from it*
- *Rebuild the exact hostname and return it*

Parameters	Type	Description
line	*Str List*	A String line listed by words

3.7.2 Returns

String

The exact hostname string with form **'name@host'**

3.8 Get_origin_connected

```
def Get_origin_connected()
```

3.8.1 Definition

Manage a ssh session with the origin server to get the raw output of the Licence request to get connected users. The used remote command is :

```
/opt/Linux_FLEXnet_Server_ver_11.16.5.1/lmutil  lmstat -a -c /opt/Linux_FLEXnet_Server_
↪ver_11.16.5.1/Licenses/Origin_20jetons.lic | grep "^.*origin\.srv-prive\.icgm\.fr/
↪27000.*"
```

3.8.2 Returns

String The output of the ssh remote command

3.9 Get_Connected

```
def Get_Connected()
```

3.9.1 Definition

Getting the full connected users list in the origin server since the ssh remote commands.

The automated ssh session request the users list to the origin server Licence manager.

Once the list stored, it is treated by regular expression to extract the hostname list.

The used regular expression to extract hostname is : **\s*[0-9A-Za-zïëîê_\s-]+**

This funtion use the **pop_double** method.

3.9.2 Returns

Str List

The full connected at Origin hostname list as Strings.

3.10 is_connected

```
def is_connected(user,Connected_content)
```

3.10.1 Definition

Check if the given user is in the connected list. The connected list is given by the **Get_Connected** method.

Parameters	Type	Description
user	*Str*	The user's hostname to test
Connected_content	*Str List*	The Connected Hostnames list

3.10.2 Returns

Boolean

The Boolean value return **True** if present **False** else

3.11 Compute_elapsed_time

```
def Compute_elapsed_time(Start_dict)
```

3.11.1 Definition

Compute the elapsed time connection dictionnary associating an user to his connexion time. The function take one parameter : the dictionnary associating to an user a string start time extracted from the Origin Licence request.

Each user is brownsed by for loop to extract hour and minuts to make the timestamp. The result is obtained making the timestamp difference between now and the start time, converted in minuts by dividing by 60.

Parameters	Type	Description
Start_dict	*Dictionnary*	The dictionnary associating to an user his start time

3.11.2 Returns

Dictionnary The dictionnary associating to an user his connexion time, computed by timestamp

3.12 Get_Connexion_Time

```
def Get_Connexion_Time()
```

3.12.1 Definition

Main algorithm manager. It is used to organize the algorithm rules :

- Get the raw connected users from origin request.
- Treating the Output via the Get_Connected method to rebuild hostname with syntax <name>@<host>
- Associate to each hostname its timestamp using following regular expression : **[0-9]+:[0-9]+**
- Compute the elapsed connexion time with the previous defined *Compute_elapsed_time* method.

It returns a dictionnary associating to each hostname its connexion time. The keys of the dictionnary should be used as list entry of connected people.

3.12.2 Returns

Dictionnary The dictionnary associating to an user his connexion time, computed by timestamp

3.13 Since version 2

This update contains following modificaton :

- **Get connexion time rewritted**
- **Merged sections** :
  - *Get Connexion Time Section*
  - *Get Connected Users Section*

The two sections have been merged to optimise the execution time storing the Conneceted users and elapsed connection time in the same dictionnary using only one ssh authentified session.

3.14 Since version 1

This is the main methods repertory needed to manage properly an Origin Server.

It contains all the following functions :

- **The Get Connexion Time Section :**
  - *Init_dict* : Initialize a dictionnary with defaults values
  - *get_max* : get the ma value from a list
  - *get_min* : Get the min value from a list
  - *is_connected* : Check if a specific user is connected
  - *build_dict* : Build the Commexion time elapsed dictionnary sort by user hostname
- **The Tftp Server Informations Getter Section :**
  - *ssh_session* : Automate an authentified ssh session
  - *Treat_out* : treat the Cisco output of an ssh session
  - *Get_Description* : Get the outlet description from a gigabithethernet socket
  - *cut_dic* : Split a dictionnary into slices to treat the parallel section
  - *get_Dict* : Get the tftp Snoop dictionnary repertoring all the needed informations
- **The Get Connected Users Section:**
  - *reverse* : Reverse the given list
  - *pop_double* : Treat the string list to retrieve hostname information
  - *Get_Connected* : Get the full hostname list from the connected users list

All these functions have been wrote for the ICGM laboratry network and must be adapted to another network (Ssh passerel identification informations, Cisco switchs name and addresses, etc…)

Please to load it directly into a Python interpreter from the command :

```
from Origin_API import *
```

CHAPTER

FOUR

TREAT_TOKENS

```
Treat_tokens.sh <mode>
```

4.1 Principe

To use the Treat_tokens.sh script, you have to already instanced the **nohup** automated script to read and analyze the logwatch file. The logwatch file must be contained into the same folder than the script.

The Treat_tokens.sh script has been writtent to automate the Tokens Management from an Origin Server log file:

- Each User take an **OUT** token to start a working session.
- Each **OUT** token will be followed by an **IN** or **OUT** token, the last emitted **IN** or **OUT** token sign the closure of the connection
- Each Token is associated to a **hostname** and a **timestamp**
- Each connected **user is managed by tokens** during his session
- Each tokens allocation and restitution is stored into the **orglabdebug.log** file

I am Dressing a *"Token map"* or *"Token array"* of the already distributed tokens.

To do so, see the followings methods :

- **Get the immediate Content of the daily logwatch file** *(generated with the same nohup script auto sheduled than the first Script)* :
  - *Get day, month and year fields from the command date*
  - *Get the line number from the split must start with command* **grep -n**
  - *Get the number of line contained in the daily logwatch with the difference between the command* **wc -l** *result and the last one*
  - *Cut the logwatch from the end with the command* **tail**
- **Get the raw Token list** associating User ID and the time field :
  - *Brownse the daily logwatch content*
  - *Filter by regular expressions line by line to get the following fields :*
    - **Tokens (IN and OUT)**
    - **Associated Hostnames (for both of them)**
    - **Exact Date-Time field**
- **Sorting tokens** by Type (IN or OUT):
  - *Converting the Date-Time field to timestamp using the command* **date -d**
  - *Switch the mode as parameter 1*
- **Treat the input** entries as a switch
  - *Differents mode filter differents results from the same list using regular expression*
- **Associate to each token an User ID** or Hostname (filtered by regular expressions):
  - *Keeping in the same order the tokens list and the hostname list*
- **Associate to each token the correct Timestamp**:
  - *From the timestamp conversion, print the correct timestamp associated to a token*

4.2 Usage

Please to use with the correct syntax :

```
./Treat_tokens.sh <mode>
```

where mode balance between :

- **1** : *Get the IN tokens*
- **2** : *Get the OUT tokens*
- **3** : *Get the IN Tokens Hostname*
- **4** : *Get the OUT Toekns Hostname*
- **5** : *Get the IN Tokens Timestamp Sorted List*
- **6** : *Get the OUT Tokens Timestamp Sorted List*

CHAPTER

FIVE

TREAT_LOG_V2.1

```
Treat_log_v2.1.sh <mode>
```

5.1 Principe

This script has been writtent to treat immediatly the logwatch file and associate to each **User ID** the correct **Ip address**.

To make it work, you have to write the logwatch file since the micro shell script and launcher. (The orglabdebug.log file manager associating a date time to an event on the orglabdebug logfile)

It is ruled by automatic script :

```
date >> ./logwatch
ss -n -t | grep 60213 >> ./logwatch
tail -n 1 /usr/local/flexlm/orglabdebug.log >> ./logwatch
```

This script is lauched periodically with commands :

```
inotifywait -q -m -e modify /usr/local/flexlm/orglabdebug.log|
while read -r filename event; do
 ./Script.sh
done
```

This script is ruled by the following algorithm :

- **Get the file logwatch** from the orgin server using the command **scp**
- **Cut and read Logwatch file** since the date fields (must be a daily Slice):
  - *Get day, month and year fields from the command* **date**
  - *Get the line number from the split must start with command* **grep -n**
  - *Get the number of line contained in the daily logwatch with the difference between the command* **wc -l** *result and the last one*
  - *Cut the logwatch from the end with the command* **tail**
- **Reading filtered content** to get the correct Informations
  - *Split the daily logwatch content from the day starting line and month starting line*
- **Filtering Ip and User** fields from Regular Expressions
  - *Associate an user variable to the following regular expression filtered data :* **[A-Za-z0-9_-êïù]+@[A-Za-z0-9_-]+**
  - *Associate an ip variable to the regular expression filtered data :* **([0-9]+\.){3}[0-9]+**
- **Associate to each User Token Event an Ip list** containing all the Inforamtions since the **ss -n -t** command output
  - *Store the differents ip address from the* **ss -n -t** *command into the IP_slice list*
  - *If the current item is an User field, associate to each user an IP list using the index count*
  - *Increment the index at each user changing*
- **For each User, stored in time, Computing the Cantor Difference between the two Ip Sets.** The result is the associated IP of the current User. In fact the first IP is immediatly avaible and permit to find the others from the principle of deduction:
  - *From the IP_list, define the first user name and the first ip address as loop starter (all the others will be deducted from the first ones)*
  - *Brownsing the array and make the absolute set difference between each stored ip address list to get the newest*

- *The newest ip address is assigned to his respective hostname and stored into the User_IP list*
- *Print the result to use it in another script as raw output*

The result is shown as a **user:ip** list association and is used in the **Get_Origin_Info_v2.1.py** to make it faster.

5.2 Usage

Please to use with the correct syntax :

```
./Treat_log_v2.1.sh
```

CHAPTER

SIX

ORIGIN_USERS_PARALLELISATION_V2

Here the pages dedied to the Pré-versions of the project.

The first one, called Origin_Users_parallelisation_v2.py use the DHCP configuration ods file to retrieve the main informations. The principe is a bit different and will be xplained function by function.

This is the main function of the algorithm used to update Origin History since log file.

This algorithm is ruled by followings steps :

- **Getting Users acount informations since the top level** : *Environnment variable getter*
- **Connecting an ssh session to the origin.srv-prive.icgm.fr server** : *Using netmiko module to automate authentified ssh session*
- **Getting raw users list Informations** : *From the output of the Origin Licence Request, Retrieve the connected users list*
- **Getting the Port Informations** : *From the* **netstat -anp** *command, retrieve the Origin server's used port number*
- **Getting the raw IP list informations** : *From the* **ss -n -t** *command, Dress the list of present IP in connexion table*
- **Getting the raw hostname list Informations** : *From the* **ss -n -t -r** *command, Get the hostname list preset in connexion table*
- **Exit the ssh session and read the Ordinateurs.ods file** : *From the Ordinateurs.ods file, Fid and store all the others needed informations as MAC @, Vlan Id, . . .*
- **Updating the Origin_history file since the newest Informations**

The results are dispayed at screen but could be write in an Origin History

CHAPTER

SEVEN

ORIGIN_USERS_PARALLELISATION_V2_API

Here you will find all the associated functions to the Origin_Users_parallelisation_v2 version.

7.1 get_Port

```
def get_Port(output)
```

7.1.1 Definition

Getting port number since the regular expressions using the output stdout from the ssh remote command output

Parameters	Type	Description
output	*String*	The ssh remote command output specified as parameter

7.1.2 Returns

Integer

The port number used by the origin server.

7.2 get_IP_list

```
def get_IP_list(IP)
```

7.2.1 Definition

Getting IP list since the regular expressions using the output stdout from the ssh remote command output The filtering operation is done in multiline mode and will be coursed match by match. The result is shown as a list of ip address.

Parameters	Type	Description
IP	*String*	The ssh remote command output specified as parameter

7.2.2 Returns

String List

The list containing all the ipaddress founded in the ssh remote command output.

7.3 get_Host_list

```
def get_Host_list(Host)
```

7.3.1 Definition

Getting Host list since the regular expressions using the output stdout from the ssh remote command output. Brownsing the string output line by line and filter each line independantly from the others to get the correct hostnames contained. The hostnames have form *name.dsi0.icgm.fr:60213*, this is the form present into the output of a **ss -n -t -r** command.

The result is the sorted list of the hostnames. To sort them, I use the **list(dict.fromkeys(liste))** command

Parameters	Type	Description
Host	*String*	The ssh remote command output specified as parameter

7.3.2 Returns

String List

A list containing all the hostnames founded into the ssh remote command output.

7.4 Read_ods

```
def Read_ods(path,Host_list,IP_list)
```

7.4.1 Definition

Reading the Ordinateurs.ods file to get associated MAC_@ & Departement ID. The Ordinateurs.ods file contain all the authorized host into the DHCP server (and so the MAC address and the Departement ID).

Parameters	Type	Description
path	*String*	Define the path of the .ods file to read
Host_list	*String List*	The given Hostname list to find values into the ods file content
IP_list	*String List*	The given IP list to link the differents informations together

7.4.2 Returns

String List

The List repertoring the following informations as item :

- Hostname
- MAC address
- Departemet ID
- IP address

7.5 ssh_session_Treat_info

```
def ssh_session_Treat_info(cisco,IPSwitchs)
```

7.5.1 Definition

Automated authentified ssh session with parameters. The associated remote command is **sh mac address-table** to automate the Cisco request y ssh.

Parameters	Type	Description
cisco	*String*	The Cisco Switch name to connect
IPSwitchs	*Dictionnary*	The dictionnary associating to each Switch name its IP address

7.5.2 Returns

String

The raw output of the ssh remote command

7.6 Treat_Info

```
def Treat_Info(Infos,IPSwitchs)
```

7.6.1 Definition

Treat Infos getted since the ods file and the ssh output both. Etablishing a link between the MAC_@ and the Cisco Socket Number. The result will be stored in a 'ready to print' list. This function is ruled by a looped algorithm :

for each cisco in the network :

- **request the associated cisco**
- **get the Cisco gigabitethernet socket** from the **sh mac address-table** output :
  - *Filter by the following regular expression* : **Gi([0-9]\/){2}[0-9]+**
- **Store the informations** with form : *'Cisco : | Vlan / Mac_@ / GiB : | Host : | Dpt : | IP_@ '*

Parameters	Type	Description
Infos	*String list*	A list containing all the needed informations linked to an user
IPSwitchs	*Dictionnary*	The dictionnary associating to each Switch name its IP address

7.6.2 Returns

String List

'Ready to print' String list where each item is associated with a user and have form : *'Cisco : | Vlan / Mac_@ / GiB : | Host : | Dpt : | IP_@ '*

7.7 Write_in_file

```
def Write_in_file(to_write,path)
```

7.7.1 Definition

Write/Update Infos in file from the path name. The Infos parameter must be with type Sorted String List as defined in the Treat_Info method.

Parameters	Type	Description
to_write	*String List*	The full content to write as defined in the treat_Info method
path	*String*	The raw path of the fle to write/update

7.8 get_Description

```
def get_Description(Data)
```

7.8.1 Definition

Updating Socket Description field and add a timestamp to the Information. To do so, I'm uing the following regular expressions :

- *Cisco socket getter* : **Gi([0-9]\/){2}[0-9]+**
- *Outlet Description getter* : **[NRJPASEP]+[0-9]+[A-K][0-9]+-[0-9]+**
- *Cisco Name getter* : **Balard-[EPACRDGH1234]+-[0-9]**

Foreach dataline in the Data list:

- Filter the two needed fields and store them in their respective variable cisco and socket
- use a ssn session to get the output of the command **show interface gigabitethernet**
- Filter the output with the Outlet Description getter expression
- Add the Description field to the dataline
- Rebuild a full Data list as result

Parameters	Type	Desccription
Data	*String List*	The String Datas as list, each dataline contain the following informations : • Cisco Name • Vlan id • MAC address • Cisco Socket • Hostname • Departemet id • IP address

7.8.2 Returns

String List

The updated Data list with description field

7.9 reverse

```
def reverse(liste)
```

7.9.1 Definition

Standard list reverse function

Parameters	Type	Description
liste	List	The list to reverse

7.9.2 Returns

List

The reversed list

7.10 get_time

```
def get_time(Data,User_rep,User_list)
```

7.10.1 Definition

Getting exact time duration since already recorded timestamp and add it to the Main Data List. This method is ruled by the followings steps:

- foreach dataline in Datas :
  - Get the IP address since regular expression filtering
  - Get the name and check if present in the User_list
  - If present, associate a timestamp
  - If the timestamp is defined, compute the difference between the now timestamp and the starting timestamp to get the Connexion time elapsed
  - Update the Data list with the Time Elapsed field

Parameters	Type	Description
Data	*String List*	The String Datas as list, each dataline contain the following informations : • Cisco Name • Vlan id • MAC address • Cisco Socket • Hostname • Departemet id • IP address • Socket Description
User_rep	*Dictionnary*	The users dictionnary extracted from the logwatch file linking to an user his strating timestamp
User_list	*Dictionnary*	The User dictionnary extracted from the logwatch file linking to an user his IP address

7.10.2 Returns

String List

The updated Data list with field Time Elapsed

7.11 treat_Users

```
def treat_Users(Users)
```

7.11.1 Definition

Managing Tokens allocation (Time Elapsed since the first Token). This method read the content of the requested Licence file.

Differents regular expressions manage the results :

- *month* : **[0-9]+\/+**
- *day* : **[^a-z]\/[0-9]+**
- *hour* : **[0-9]+\:**
- *minuts* : **\:[0-9]+**
- *user* : **^\s*[^:\s]+**
- *PC* : **[A-Z0-9]+-[A-Z0-9]+**

Once the differents fields retireved from regular expressions, the return dictionnary is populated with users name and the linked timestamp.

Parameters	Type	Description
Users	*String*	The output of the Origin Licence Request to get Connected users

7.11.2 Returns

Dictionnary

The dictionnary associating to an user name its connexion starting timestamp

7.12 cut_dic

```
def cut_dic(Cisco_Dic,div)
```

7.12.1 Definition

Split Dictionnary into div differents dictionnary to treat them with parallelism.

Parameters	Type	Description
Cisco_Dic	*Dictionnary*	The Cisco 2 Ip main dictionnary
div	*Integer*	The number of dictionnary slice needed

7.12.2 Returns

Dictionnary List

A list af *div* differents dictiononary

7.13 Cut_log

```
def Cut_log()
```

7.13.1 Definition

Cut logfile since the date (today as default). The logwatch file is primary stored into the local folder. Once done, It cut the logwatch file since the today date.

It write the daily logwatch content instead of your local copy of the logwatch file.

7.14 read_log

```
def read_log(path)
```

7.14.1 Definition

Read the log file and filter the content by regular expression to get the main content of the logwatch file.

Parameters	Type	Description
path	*String*	The path where the logwatch is stored

7.14.2 Returns

List of List

The list of list containing the main content sorrted by token in order

7.15 Treat_log

```
def Treat_log(match_list)
```

7.15.1 Definition

Treat Log file content since regular expression to get

- *IP_@ list* : **([0-9]+\.)+[0-9]+**
- *New user information* : **[A-Za-zëùî0-9]+@[A-Z0-9]+-[A-Z0-9]+**

The content analized is the outputof the read_log method sorted by token. This function link an user to an ip list. This ip list contain all the suceptible ip for this user.

Parameters	Type	Description
match_list	*List of List*	The list of list containing the main content sorrted by token in order

7.15.2 Returns

Dictionnary

A dictionnary associating to an user name the associated ip address list from the logwatch file content

7.16 diff_list

```
def diff_list(l1,l2)
```

7.16.1 Definition

Compute difference between 2 lists to get the most suceptible ip to assign.

The difference between **two set A and B (A-B)** give us the ip addresses present in **A** but **NOT in B**.

Parameters	Type	Description
l1	*List*	An Ip list extracted from the Treat_log method return dictionnary
l2	*List*	An Ip list extracted from the Treat_log method return dictionnary

7.16.2 Returns

String List

The list containing the difference between l1 and l2

7.17 Diff_log

```
def Diff_log(User_dic)
```

7.17.1 Definition

Associate a new user to the difference between 2 log slice. Results will be stored into a python dictionnary.

This function is ruled by the following instructions :

- **Brownsing the User_dic dictionnary** and filter the hostname by regular expression
- **Computing the difference between two adjacents lists** using the diff_list function
- **Associate to an user name its own ip addresses set**

Parameters	Type	Description
User_dic	*Dictionnary*	The dictionnary associating to an user name the associated ip address list from the logwatch file content from the Treat_log function

7.17.2 Returns

Dictionnary

The dictionnary associating to an user name an ip addresses set.

7.18 Treat_diff

```
def Treat_diff(User_dic)
```

7.18.1 Definition

Compute the **Set difference by User ID** between two sets of ip address to get the correct one.

In fact treat the output of the **Diff_log** function (removing indexes and merge list if necessary)

Parameters	Type	Description
User_dic	*Dictionnary*	The dictionnary associating to an user name an ip addresses set from the Diff_log function

7.18.2 Returns

Dictionnary

The updated dictionnary associating to an user name an ip addresses

7.19 get_max

```
def get_max(liste)
```

7.19.1 Definition

Get the max value's index of the list.

Parameters	Type	Description
liste	*List*	Integer or Float list to treat

7.19.2 Returns

Integer / Float

The index of the maximum value of the list

7.20 get_ip

```
def get_ip(User_dic,IP_list)
```

7.20.1 Definition

Get the real (most susceptible one) IP_@ from an user name using successives reults from functions :

- **read_log**
- **Treat_log**
- **Diff_log**
- **Treat_diff**

The favorite IP is choosen by number of appearence into the merged list of suceptibles ip address from difference.

Parameters	Type	Description
User_dic	*Dictionnary*	The dictionnary from the successive intermediate functions associating an user a merged list of candidates
IP_list	*String List*	The IP list of connected users

7.20.2 Returns

Dictionnary

The Final dictionnary associating to an user the most suceptible IP address from logwatch analyze

7.21 diff_ip

```
def diff_ip(ipA,ipB)
```

7.21.1 Definition

Get the raw difference between 2 ip address.

Exemple :

- *IP_a* =10.14.20.1
- *IP_b* =10.14.2 **1.3**

The difference will be **1.3**

Parameters	Type	Description
ipA	*String*	IP address to compare
ipB	*String*	IP address to compare

7.21.2 Returns

String

The raw difference between both of the ip address

7.22 get_IP_from_log

```
def get_IP_from_log(IP_list)
```

7.22.1 Definition

DHCP data finder Main Resolution Algorithm. This algorithm use and manage the functions:

- **read_log**
- **Treat_log**
- **Diff_log**
- **Treat_diff**
- **get_ip**
- **diff_ip**

It restore the final dictionnary associating to an user its ip address.

Parameters	Type	Description
IP_list	*String List*	The list extracted from the command's output **ss -n -t**

7.22.2 Returns

Dictionnary

The Final dictionnary associating to an user the most suceptible IP address from logwatch analyze

CHAPTER

EIGHT

GET_USER_INFO_FROM_IP_V2

This script is the optimised version of the Origin_Users.py script.

It uses the **Treat_log_v2.sh** file to get an immediate association between user ID and their IP.

Since the two first versions, the optimised version is ruled differently from the first one :

- **Cut logfile** since the date (today as default)
- **Read and extract informations** from the logwatch file with the associated *Treat_log_v2.sh* Scripot
- **Open, read & Treat the logwatch** file :
  - **Getting IP list** associated to a timed & named token. The resultys are stored by time order, arbitrary indexed from 1 -> n
  - **Getting host ID** from the full Origin user name (with form name@host) => Allow multiple users sessions on the same host
  - **Compute the Cantor difference** between two adjacents set (indexed +- 1) to get the User's associated IP
- **Building DHCP dictionnary** and get infos since the given IP adresses list as parameter :
  - **Building DHCP Dictionnary**
  - **Updating Users Dictionnary** since the DHCP dictionnary from the ip correspondance (as key entry of the Users dictionnary)
  - **Updating the Users Dictionnary** since the Cisco output command : ssh <Cisco_name> 'show mac address' to get the associated cisco switch ID and the gigabit ethernet ID
- **Finaly write the RAM stored informations dictionnary** into the Origin_history file

Please to use with the correct syntax :

```
python3 Get_User_Info_From_IP_v3.py
```

The script must be used into an equivalent environment structure :

```
.
├── DHCP
│   └── Get_User_Info_From_IP_v3.py
└── dhcpd-vlan_i.conf
└── dhcpd-vlan_i+1.conf
.
.
.
└── dhcpd-vlan_n.conf
```

The result is shown with the following syntax :

```
{'mac': '90b1.1ca3.3575', 'ip': '10.14.18.145', 'hostname': '"BBBAACCC"', 'departement':
↪'DPT4', 'vlan': 513, 'cisco': 'Balard-PAC-2', 'socket': '1/0/36', 'Description':
↪'RJLG07-01', 'origin_name': 'c2mstud@c2mstud3-pc', 'connexion time': '198.
↪3088238040606 min'}
```

With :

Field Identifier	Data Type	Description
mac	*Hexadecimal string*	*The full mac address of the current User*
ip	*Decimal string*	*The full fixed IP from the origin server*
hostname	*String*	*The Hostname from the DHCP server (could be different from the Origin server Hostname)*
departement	*String*	*The departement description section*
vlan	*Integer*	*The sub-network lan Identifier*
cisco	*String*	*The Cisco Switch Identifier Name*
socket	*Decimal String*	*The associated Gigabit Ethernet socket (with form **x/y/z*)**
Description	*String*	*The associated outlet exact name (as it is written in a Cisco Switch)*
origin_name	*String*	*The Origin User's avatar name*
connexion time	*Float*	*If still connected, the connection time of the User, else the starting connection time*

Finally written into the Origin_history file into the **origin.srv-prive.icgm.fr** server.

CHAPTER

NINE

GET_USER_INFO_FROM_IP_V2_API

Here you will find all the associated functions to the Get_User_Info_From_IP_v2_API version.

9.1 cut_dic

```
def cut_dic(Cisco_Dic,div)
```

9.1.1 Definition

Split Dictionnary into div differents dictionnary. Useful function for parallelism section.

Parameters	Type	Description
Cisco_Dic	*Dictionnary*	The dictionnary containing all the Cisco name:IP address informations
div	*Integer*	The number of slice to get

9.1.2 Returns

List of Dictionnaries The List of Sliced dictionnaries

9.2 Del_Duplicate

```
def Del_Duplicate(liste)
```

9.2.1 Definition

Utilitary function removing all the duplicated values of the given list.

Parameters	Type	Description
liste	*List*	The list to treat

9.2.2 Returns

List The same list without duplicated values

9.3 ssh_session

```
def ssh_session(cisco,command)
```

9.3.1 Definition

Treat a ssh remote command session. Automate the authentified ssh connexion and restitute the remote command output

Parameters	Type	Description
cisco	*String*	The cisco switch name to connect
command	*String List*	The command list as a string list

9.3.2 Returns

String The remote command output

9.4 Get_Users_Info

```
def Get_Users_Info(IP_list)
```

9.4.1 Definition

Building DHCP dictionnary and get infos since the given IP adresses list as parameter.

To do so, the DHCP dictionnary construction obey to the following looped instructions :

- **Define the following regular expressions** to retrieve the differents fields from the DHCP configuration files :
  - MAC address : **([0-9A-Fa-f]{2}\:){5}[0-9A-Fa-f]{2}**
  - IP address : **fixed.***
  - Raw ip : **([0-9]+\.){3}[0-9]+**
  - Hostname : **\"[A-Za-z0-9-_]+\"**
  - Cisco name : **Gi([0-9]+\/){2}[0-9]+**
  - Outlet Description : **[NRJPASEP]+[0-9]+[A-K][0-9]+-[0-9]+**
- **For each vlan present on the network** :
  - **Read the associated dhcps-vlanId.conf file**
  - **For each slice of the file** :
    - *Get informations since the regular expressions filtering*
    - *Store them into the tmp_dict dictionnary*
    - *Append the dictionnary to the Users list*
  - **Delete duplicated values** from the Users list if necessary
  - **Store the Users list into the DHCP_Dict dictionnary** (sorted by Vlan name)

The main algorithm used to link informations together is ruled by the followings steps :

- **Regular Expressions Definition**
- **Building DHCP Dictionnary**
- **Updating Users Dictionnary since the DHCP dictionnary** *from the ip correspondance (as key entry of the Users dictionnary)*
- **Updating the Users Dictionnary since the Cisco output command** : **ssh <Cisco_name> 'show mac address'** *to get the associated cisco switch ID and the gigabit ethernet ID*

Parameters	Type	Description
IP_list	*String List*	The ip address list to treat as input (corresponding to the connected users list)

9.4.2 Returns

Dictionnary The Users Dictionnary repertoring all the needed informations from the DHCP configuration files.

9.5 Cut_log

```
def Cut_log()
```

9.5.1 Definition

Cut logfile since the date (today as default) and write it in the local folder

This function cut the logwatch file since the current date and restitute the content until the end of file.

9.6 time_to_timestamp

```
def time_to_timestamp(str_time)
```

9.6.1 Definition

Utilitary converter function getting timestamp from the given string date. It uses regular expressions filtering to get time and month field. The timestamp is generated since the retrieved informations passed as parameters of the **time.mktime()** command.

Parameters	Type	Description
str_time	*String*	The date-time formated as String

9.6.2 Returns

Integer The correct converted timestamp

9.7 Read_and_treat_log

```
def Read_and_treat_log(path)
```

9.7.1 Definition

This function make the association between a Origin user name and its own Ip address. Read and extract the following informations from the logwatch file using regular expressions :

- Date : **[a-z]+([^a-z]+.*[0-9]*\n)+**
- IP : **([0-9]+\.)+[0-9]+**
- Name : **\"OriginPro\".**
- PC : **\@.***
- Time : **^[a-z].***

To get informations from the logwatch file, the algorithm is ruled by instructions :

- **Regular Expression Definition**
- **Variables Definition**
- **Open and read the logwatch file**
- **Getting IP list associated to a timed & named token.** The results are stored by time order, arbitrary indexed from 1 -> n
- **Getting host ID from the full Origin user name** (with form name@host) => Allow multiple users sessions on the same host
- **Compute the Set difference between two adjacents Ip set** (indexed +- 1) to get the User's associated IP

Parameters	Type	Description
path	*String*	The path to the logwatch file to read

9.7.2 Returns

Dictionnary The name_ip_dict dictionnary associating to an Origin user name its own ip address

9.8 Write_in_file

```
def Write_in_file(to_write,path)
```

9.8.1 Definition

Write Infos in file in append mode : if file already exists, the content is added from the end.

Parameters	Type	Description
to_write	*String List*	The content to write as String List to write line by line
path	*String*	The path where the file must be written

9.9 get_Connected

```
def get_Connected()
```

9.9.1 Definition

Get connected user list since the orgin token licence. The function read the Licence manager from Origin and extract the connected users name list. This function use regular expression matching to get user name from the ssh remote command output :

```
/opt/Linux_FLEXnet_Server_ver_11.16.5.1/lmutil  lmstat -a -c /opt/Linux_FLEXnet_Server_
↪ver_11.16.5.1/Licenses/Origin_20jetons.lic | grep "^.*origin\.srv-prive\.icgm\.fr/
↪27000.*"
```

9.9.2 Returns

String List The list containing all the Origin connected hostnames

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search