Origin_Server

Release 2.1

CABOS Matthieu

Jan 12, 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

CHAPTER

ONE

GET_ORIGIN_INFO.PY

1.1 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.

1.2 Version 2

That version is similar to the first one. There is no display in this one but the Origin History file is properly written.

Please to use with the correct following syntax :

```
python3 Get_Origin_Info_v2.py
```

1.3 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

TWO

ORIGIN_API

Author *CABOS Matthieu*

Date *2021/2022*

Organization *ICGM-CNRS*

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 *
```

2.1 Init_dict

```
def Init_dict(Hostname,flag)
```

2.1.1 Definition

Intialisation of Dictionnaries with default values function.

Parameters	Type	Description
Hostname	*Str List*	A list repertoring the hostnames of users
flag	*Bool*	**The Boolean Flag to define the mode between:** • Max finder initialization • Min finder initialization

2.1.2 Returns

Dictionnary

The dictionnary associating to an User hostname its default value

2.2 get_max

```
def get_max(liste)
```

2.2.1 Definition

Get the maximum value from the given list.

Parameters	Type	Description
liste	*Int List*	The list to analyze

2.2.2 Returns

Integer

The maximum value of the list.

2.3 get_min

```
def get_min(liste)
```

2.3.1 Definition

Get the minimum value from the given list.

Parameters	Type	Description
liste	*Int List*	The list to analyze

2.3.2 Returns

Integer

The minimum value of the list.

2.4 is_connected

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

2.4.1 Definition

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

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

2.4.2 Returns

Boolean

The Boolean value return True if present False else

2.5 build_dict

```
def build_dict()
```

2.5.1 Definition

Building Timing dictionnary from the logwatch file.

To do so, I use the treat_tokens.sh script file with the following arguments 3,4,5,6 to get respectively the following informations :

- **IN Tokens Hostname**
- **OUT Token Hostname**
- **IN Tokens Timestamp allocation**
- **OUT Tokens Timestamp allocation**

The builder algorithm is defined as following :

- **Once informations retrieved** from Treat_tokens script, **transtype** them to python list and merge IN and OUT hostname lists and IN and OUT Timestamp Lists.
- **For each user found in the Hostname list, store the associated timestamps** into a temporary list and associate the list to a user name via Python Dictionnary Token_dict.
- Brownsing the Token_dict and **for each hostname present in the Connected Users List, Compute the absolute time value.**

The Final reult is given as a dictionnary associating to each connected user hostname its connection time elapsed.

2.5.2 Returns

Dictionnary

The Timer dictionnary associating to a hostname its connection time

2.6 ssh_session

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

2.6.1 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

2.6.2 Returns

Dictionnary

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

2.7 Treat_out

```
def Treat_out(output)
```

2.7.1 Definition

Treating shell command ouptut since the tftp Boot informations reading.

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

- **regex_ip**
- **regex_mac**
- **regex_socket**
- **regex_vlan**
- **regex_switch**

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

2.7.2 Returns

Dictionnary

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

2.8 Get_Description

```
def Get_Description(Snoop_Dict)
```

2.8.1 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

2.8.2 Returns

Dictionnary

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

2.9 cut_dic

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

2.9.1 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

2.9.2 Returns

List

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

2.10 get_Dict

```
def get_Dict()
```

2.10.1 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.

2.10.2 Returns

Dictionnary

The Snoop dictionnary repertoring all the needed network informations.

2.11 reverse

```
def reverse(line)
```

2.11.1 Definition

Perso reverse list function

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

2.11.2 Returns

Str List

The reversed list

2.12 pop_double

```
def pop_double(line)
```

2.12.1 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

2.12.2 Returns

String

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

2.13 Get_Connected

```
def Get_Connected()
```

2.13.1 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.

This funtion use the pop_double method.

2.13.2 Returns

Str List

The full connected at Origin hostname list as Strings.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search