Cisco Mapping

Release 2

CABOS Matthieu

Oct 14, 2021

CONTENTS:

CHAPTER

ONE

CISCO_MAPPING_ALGORITHM

1.1 Algorithm

Welcome to the Cisco Mapping Source Code documentation. This code has been provided to manage informations from Cisco Switch (as number of connected people, with their associated informations). It should be used to Administrate a DHCP server using Tftpboot to store Connected Users informations. A DHCP server is ruled by mac adresses, each fixed ip adress have one and only one associated mac adress. The DHCP server provide to Authorised Users the full network access. A non-authorised hostname won't get any access on the network.

The used algorithm is ruled by the followings steps :

- **Getting infos from the Tftpboot server** : We get the stored informations since the Tftp server (stored in the /var/lib/tftpboot/snoop/ folder)
- **Building IP:MAC dict** : We build the dictionnary of the ip adresses associated to the hardware mac adresses of the connected users
- **Brownsing ods file** : We read the Configuration ods file containing Hostnames, Mac adresses, Vlan informations, and comments
- **Searching current mac in @MAC database and updating Dictionnary Fields** : We populate the Final dictionnary with the form : Hostname | @mac | Vlan Id | @ip | switch name | switch @ip | Port number | Switch Gigabit informations | Socket Number | Comments
- **Updating Comments Field** : We update the Comments Fields from the Configuration ods file
- **Updating Room Sockets Names Field** : We update The Socket Number fields using the Cisco *show interface description* command
- **Building the not-conected Dictionnarry** : We build as a second Sheet the non-connected authorised Users of the DHCP Server
- **Packaging as array to write** : We package these arrays to be wrote into an ods file
- **Saving ods file** : We save the Raw Content generated by the script into the output ods file.

All these steps of the algorithm have been released using the following functions. Each of these one have been explained into a specific paragraph.

1.2 Source Code

```
switch_dict={
'balard-1D-1':'10.14.0.49',
'balard-1G-1':'10.14.0.51',
'balard-2D-1':'10.14.0.58',
'balard-2G-1':'10.14.0.60',
'balard-2H-1':'10.14.0.62',
'balard-3D-1':'10.14.0.67',
'balard-3G-1':'10.14.0.69',
'balard-3G-2':'10.14.0.70',
'balard-4C-1':'10.14.0.74',
'balard-4D-1':'10.14.0.76',
'balard-4G-1':'10.14.0.78',
'balard-4H-1':'10.14.0.80'
}

switch_dict2={
'10.14.0.49':'Balard-1D-1',
'10.14.0.51':'Balard-1G-1',
'10.14.0.58':'Balard-2D-1',
'10.14.0.60':'Balard-2G-1',
'10.14.0.62':'Balard-2H-1',
'10.14.0.67':'Balard-3D-1',
'10.14.0.69':'Balard-3G-1',
'10.14.0.70':'Balard-3G-2',
'10.14.0.74':'Balard-4C-1',
'10.14.0.76':'Balard-4D-1',
'10.14.0.78':'Balard-4G-1',
'10.14.0.80':'Balard-4H-1'
}


# Getting infos from the Tftpboot server
os.system('scp mcabos@tftp.srv-prive.icgm.fr:/var/lib/tftpboot/snoop/* .')
Dpt_dict=Get_Dpt('Ordinateurs.ods')

#Building IP:MAC dict
ip2mac={}
for switch in switch_dict.keys():
        Content=get_content(switch)
        ip2mac[switch]=build_ip_mac_dict(Content)

# Brownsing ods file
file_name='Ordinateurs.ods'
records = p.get_array(file_name=file_name)
regex=r"/[0-9]+$"
Final_dict={}
Final_dict['Nom de la machine']=['@mac','Departement', '@ip machine', 'nom switch', '@ip␣
↪switch', 'n° port', 'Triolet Gigabit','n° Prise','Commentaires']

# Searching current mac in @MAC database and updating Dictionnary Fields
```

(continues on next page)

(continued from previous page)

```
for record in records:
        for switch in switch_dict.keys():
                for k,v in ip2mac[switch].items():
                        if record[1] == k :
                                matches=re.finditer(regex,v[1],re.MULTILINE)
                                for matchNum, match in enumerate(matches, start=1):
                                        port=match.group()[1:]
                                Final_dict[record[0]]=[k,Dpt_dict[record[0]],v[0],switch,
↪switch_dict[switch],port,"Gi"+v[1],"",'']

# Updating Comments Field
Comm=Get_Comm('Ordinateurs.ods',Final_dict)
for k,v in Final_dict.items():
        if not (k == 'Nom de la machine'):
                tmp=v
                tmp[8]=Comm[k]
                Final_dict[k]=tmp

# for sw in liste_switch:
#       Final_dict=update_Room_Sockets(sw,Final_dict)

# Updating Room Sockets Names Field
for Cisco_name in switch_dict2.values():
        Final_dict=Cis2Socket(Cisco_name,Final_dict)

# Building the not-conected Dictionnarry
Not_Conctd_Dict=Get_not_connected_dict('Ordinateurs.ods',Final_dict)

# Packaging as array to write
line=[]
to_write=[]
for k,v in Final_dict.items():
        line=[]
        line.append(k)
        line.extend(v)
        to_write.append(line)

to_write_ntc=[['Nom de la machine','@mac','Departement', '@ip machine', 'nom switch',
↪'@ip switch', 'n° port', 'Triolet Gigabit','n° Prise','Commentaires']]
for k,v in Not_Conctd_Dict.items():
        line=[]
        line.append(k)
        line.extend(v)
        to_write_ntc.append(line)

Content={'Sheet 1':to_write, 'Sheet2':to_write_ntc}
# print(Content)
# Saving ods file
book = p.Book(Content)
book.save_as('TftpBoot_List.ods')

# p.isave_as(array=to_write,dest_file_name='TftpBoot_List.ods')
```

(continues on next page)

(continued from previous page)

```
os.system('rm Description*')
os.system('rm balard*')
```

CHAPTER

TWO

FUNCTIONS

2.1 build_ip_mac_dict

```
def build_ip_mac_dict(tftp_Content):
```

2.1.1 Algorithm

Building Ip 2 @Mac dictionnarry from tftp boot server files (connected people). We are getting the full connected Users Mac => IP dictionnary using regular expression :

- **[0-9a-z]{4}\.[0-9a-z]{4}\.[0-9a-z]{4}** : Give us the MAC adress since the tftpboot files
- **([0-9]\/){2}[0-9]*** : Give us the Hardware Cisco Port Number since the tftpboot files
- **\d+\.\d+\.\d+\.\d+** : Give us the IP Adress since the tftpboot files

Parameters	Type	Description
tftp_Content	string	The tftpboot file raw content

Returns Dictionnary : The dictionnary with ip/mac correspondance

---

2.1.2 Source Code

```
ip2mac={}
MAC=""
regex = r"[0-9a-z]{4}\.[0-9a-z]{4}\.[0-9a-z]{4}"
regex2=r"([0-9]\/){2}[0-9]*"
ip=re.compile(r'\d+\.\d+\.\d+\.\d+')
for line in tftp_Content:
        matches = re.finditer(regex, line, re.MULTILINE)
        res_ip=ip.match(line)
        for matchNum, match in enumerate(matches, start=1):
                if (not res_ip == None):
                        MAC=match.group()[0:2]+":"+match.group()[2:4]+":"+match.
↪group()[5:7]+":"+match.group()[7:9]+":"+match.group()[10:12]+":"+match.group()[12:14]
        matches = re.finditer(regex2, line, re.MULTILINE)
        for matchNum, match in enumerate(matches, start=1):
                GiPort=str(match.group())
                if (not res_ip == None):
                        ip2mac[MAC]=(res_ip.group(0),GiPort)
return ip2mac
```

2.2 get_content

```
def get_content(switch_name):
```

2.2.1 Algorithm

Get content from file since the switch_name argument. This function read the file and store informations into the return value.

Parameters	Type	Description
switch_name	String	The exact switch_name from switch_dict keys

Returns String : The full Content of the file stored into a String Variable

2.2.2 Source Code

```
f=open(switch_name,'r')
return f.readlines()
```

2.3 write_in_tmp

```
def write_in_tmp(ip_switch):
```

2.3.1 Algorithm

Get SNMP informations and store it into the tmp file.

Parameters	Type	Description
ip_switch	String	The exact IP adress of the current switch

Returns None

2.3.2 Source Code

```
os.system('snmpwalk -v 1 -c comaccess '+str(ip_switch)+':161 1.3.6.1.2.1.2.2.1.6 > tmp
↪'+str(ip_switch))
```

2.4 Get_switch_port_dict

```
def Get_switch_port_dict(ip_switch):
```

2.4.1 Algorithm

Read the tmp file containing SNMP informations and sort and store them into a Dictionnary with form : @Mac : Hardware Port Number

Parameters	Type	Description
ip_switch	String	The exact IP adress of the current switch

Returns Dictionnary : The dictionnary associating a @mac to the hardware port number

2.4.2 Source Code

```
liste_addr=os.popen('cat tmp'+str(ip_switch))
regex = r"([0-9a-z]{2}:){5}[0-9a-z]{2}"
regex2=r"\.[0-9]+"
Switch_port_dict={}

for line in liste_addr.readlines():
        current_mac=""
        current_port=""
        matches = re.finditer(regex, line, re.MULTILINE)
        matches2 = re.finditer(regex2, line, re.MULTILINE)

        for matchNum, match in enumerate(matches, start=1):    # Looking for @MAC
                current_mac=str(match.group())
        for matchNum, match in enumerate(matches2, start=1):    # Looking for␣
↪corresponding port number
                current_port=str(match.group())
                if current_mac :
                        Switch_port_dict[current_mac]=current_port[1:]

return Switch_port_dict
```

2.5 Get_Port_and_GB

```
def Get_Port_and_GB(ip_switch,Final_dict)
```

2.5.1 Algorithm

Populate the Final Dictionnary with Hardware Port Number values from Cisco SNMP Values (as verification of configuration...).

Parameters	Type	Description
ip_switch	String	The exact IP adress of the current switch
Final_dict	Dictionnary	The Final Dictionnary to be updated

Returns Dictionnary : The Final Dictionnary to be write updated

2.5.2 Source Code

```
Switch_port_dict=Get_switch_port_dict(ip_switch) # As example
liste_addr=[]

# Gettind @mac list
for k in Switch_port_dict.keys():
        liste_addr.append(k)

# Populate the Final Dictionnary to be write
for k,v in Final_dict.items():
        if (v[0] in liste_addr):
                v[4]=Switch_port_dict[v[0]]
                v[5]=os.popen('./Cisco.sh '+str(v[3])+' 1 '+str(v[4])).read()
        pass

return Final_dict
```

2.6 Cisco2Socket

```
def Cisco2Socket(Cisco_name,*args)
```

2.6.1 Algorithm

Getting the exact Room Socket Name from the GigabitEthernet Triolet provided by Cisco informations.

Parameters	Type	Description
Cisco_name	String	The exact name of the Switch
args	String	A long string containing all the Hardware Cisco Port Number separated with a space key

Returns List : A List containing all the Room Socket Exact Name

2.6.2 Source Code

```
Socket_name=[]
for i in range(len(args)):
        Socket_name.append(args[i])

Cisco_list=['Balard-EP-1','Balard-PAC-1','Balard-PAC-2','Balard-RDC-1','Balard-1C-1',
↪'Balard-1D-1','Balard-1G-1','Balard-1G-2','Balard-1H-1','Balard-2C-1','Balard-2D-1',
↪'Balard-2G-1','Balard-2H-1','Balard-2H-2','Balard-3C-1','Balard-3D-1','Balard-3G-1',
↪'Balard-3G-2','Balard-3H-1','Balard-4C-1','Balard-4D-1','Balard-4G-1','Balard-4H-1']
f=open("Cisco2Socket.sh","a")
f.write('#!/bin/bash\n# Author : CABOS Matthieu\n# Date : 08/10/2021\nterm shell\n')
Cisco_Rep=[]
res={}

for i in range(1,4):
        for j in range(1,49):
                f.write('show interface GigabitEthernet'+str(i)+'/0/'+str(j)+' | grep
↪"N[0-9][A-Z][0-9][0-9]*-[0-9]*" \n')

f.write('show interface GigabitEthernet0/0/0')
f.close()
os.system('ssh '+str(Cisco_name)+" < Cisco2Socket.sh > tmp2.txt")
os.system('grep -v "^[[:space:]]*$" tmp2.txt > tmp2')
os.system('rm tmp2.txt')
i=7
nb_ligne=int(os.popen('wc -l tmp2 | cut -d " " -f1').read())-i
ind=1
jnd=1
while i <= nb_ligne:
        res[str(ind)+'/0/'+str(jnd)] = os.popen('cat tmp2 | head -'+str(i)+' | tail -2 |␣
↪grep "N[0-9][A-Z][0-9][0-9]*-[0-9]*" | cut -d " " -f4 | sed "s/,//"').read()
        i+=2
```

(continues on next page)

(continued from previous page)

```
        jnd+=1
        if jnd==49:
                ind=(ind + 1) if (ind <= 4) else 1
                jnd=1
os.system('rm tmp2')
os.system('rm Cisco2Socket.sh')
rez=[]
GBname=''
for socket in Socket_name :
        for k,v in res.items():
                if k==socket:
                        GBname=v
                        break
        rez.append(GBname[:-1])
return rez
```

2.7 Cis2Socket

```
def Cis2Socket(Cisco_name,Final_dict)
```

2.7.1 Algorithm

Getting the exact Room Socket Name from the GigabitEthernet Triolet provided by Cisco informations. The Socket Name is stored in the given Dictionnary

Parameters	Type	Description
Cisco_name	String	The exact name of the Switch
Final_dict	Dictionnary	The Final Dictionnary to be updated

Returns Dictionnary : The Final Dictionnary to be write updated

2.7.2 Source Code

```
regex=r'[A-Z][0-9][A-Z][0-9]+.[0-9]*'
os.system('ssh -t '+str(Cisco_name)+' show interface description > Description_
↪'+str(Cisco_name))
f=open('Description_'+str(Cisco_name),'r')
l=f.readlines()
Tmp_dict={v:k for k,v in switch_dict2.items()}
Plug_liste=[]

for k,v in Final_dict.items():
        if v[4] == Tmp_dict[Cisco_name]:
                Plug_liste.append((v[6][2:],k))

for line in l:
        for hw in Plug_liste:
            if hw[0] in line:
                matches = re.finditer(regex, line,re.MULTILINE)
                for matchNum, match in enumerate(matches, start=1):
                        tmp=Final_dict[hw[1]]
                        tmp[7]=match.group()
                        Final_dict[hw[1]]=tmp
                del Plug_liste[Plug_liste.index(hw)]

return Final_dict
```

2.8 update_Room_Sockets

```
def update_Room_Sockets(ip_switch,Final_dict)
```

2.8.1 Algorithm

Updating the Room Sockets Name field of the Dictionnary using the Cisco2Socket Procedure. Each Switch will be treated **independantly** from each others. It must be applied to each Switch to get the full Contents updated.

Parameters	Type	Description
ip_switch	String	The exact IP adress of the current switch
Final_dict	Dictionnary	The Final Dictionnary to be updated

Returns Dictionnary : The updated Dictionnary

2.8.2 Source Code

```
liste=[]
Plug_liste=[]
for k,v in Final_dict.items():
        if v[3]==ip_switch:
                liste.append(v[5])
ind=0
Plug_liste=Cisco2Socket(switch_dict2[ip_switch],' '.join(liste))
for k,v in Final_dict.items():
        if v[3]==ip_switch:
                try:
                        v[6]=Plug_liste[ind]
                        ind+=1
                except:
                        break

return Final_dict
```

2.9 Get_Dpt

```
def Get_Dpt(file_name)
```

2.9.1 Algorithm

Getting Departement ID from the file_name ods file containing all the Vlans Informations. The Vlan name is readed and associated to its Id number.

Parameters	Type	Description
file_name	String	An .ods file to read

Returns Dictionnary : The Departement dictionnary associating a Departement Id to a Computer Name on the network

2.9.2 Source Code

```
Dpt2Int_dict={
'DPT1':510,
'DPT2':511,
'DPT3':512,
'DPT4':513,
'DPT5':514,
'SGAF':524,
'INSTRU-ON':515,
'SSI':525,
'INSTRU-OFF':516,
'IMPRIM':518,
'IDRAC-CIN':528,
'IDRAC':501,
'ExpProtect':526
}

Dpt_dict={}
Records = p.get_array(file_name=file_name)
Dpt_name=''

for record in Records:
        for Dpt,v in Dpt2Int_dict.items():
                if Dpt in record[2] :
                        Dpt_dict[record[0]]=v
                        break

return Dpt_dict
```

2.10 Get_Comm

```
def Get_Comm(file_name,Final_dict)
```

2.10.1 Algorithm

Getting Comments fields from the '.ods' file_name. It returns a Dictionnary associating a Computer name to its Comments.

Parameters	Type	Description
file_name	String	The .ods file_name to read
Final_dict	Dictionnary	The Main Informations Dictionnary to read

Returns Dictionnary : A Dictionnary associating Comments to the linked Computer Name on the network

2.10.2 Source Code

```
Comm_dict={}
Records = p.get_array(file_name=file_name)
Comm=''

for record in Records:
        if record[0] in Final_dict.keys():
                Comm_dict[record[0]]=record[3]
return Comm_dict
```

2.11 Get_not_connected_dict

```
def Get_not_connected_dict(file_name,Final_dict)
```

2.11.1 Algorithm

Similary building a Dictionnary with fewer informations for the disconnected Users.

Parameters	Type	Description
file_name	String	The .ods file_name to read
Final_dict	Dictionnary	The Main Informations Dictionnary to read

Returns Dictionnary : A Dictionnary linking informations from the server to store the Disconnected Authorised Users

2.11.2 Source Code

```
Not_Conctd_Dict={}
Records = p.get_array(file_name=file_name)
dpt_dict=Get_Dpt(file_name)

for record in Records:
        if not record[0] in Final_dict.keys():
                try:
                        Not_Conctd_Dict[record[0]]=[record[1],Dpt_dict[record[0]],'','','
↪','','','',record[3]]
                except:
                        pass
return Not_Conctd_Dict
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search