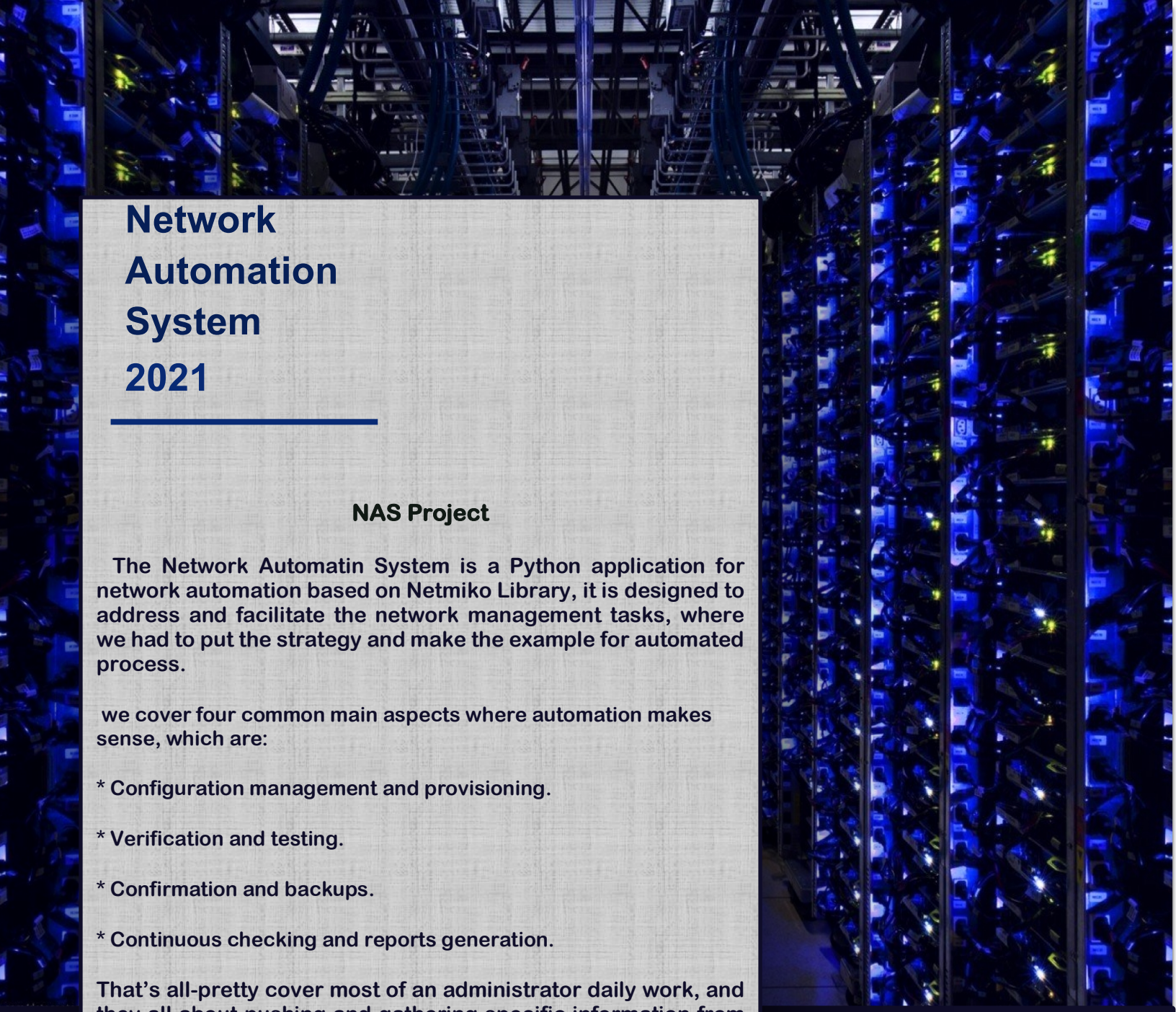


Network Automation System 2021

NAS Project

The Network Automatin System is a Python application for network automation based on Netmiko Library, it is designed to address and facilitate the network management tasks, where we had to put the strategy and make the example for automated process.

we cover four common main aspects where automation makes sense, which are:

* Configuration management and provisioning.

* Verification and testing.

* Confirmation and backups.

* Continuous checking and reports generation.

That's all-pretty cover most of an administrator daily work, and they all about pushing and gathering specific information from devices CLI using SSH protocol.

MAY 25

Team name: NETWORKLAND™
Authored by: **Nouhaila**

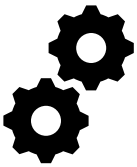
directed by:

YASSINE & NOUHAILA

Summary:

First thought

(Inspiration, Project-selection …)



Working program

(Tools, Blueprint, Arranging, Team …)

Program organization

(Roles of functions & Program explanations)

Qualitative assessment of the work

(Difficulties, Screenshots, execution scenarios …)

Conclusion

(Overview, Special thanks)

First thought

Since the beginning of this year, we have been thinking about our graduation project, we were very confused between several topics,

⇒ **Password craking with Kali-Linux and hashcat**

⇒ **Setup a VPN server in the Cloud**

⇒ **Setup a VOIP server in the Cloud**

⇒ **IOT smart home**

⇒ **Raspberry Pi Firewall and Intrusion Detection System**

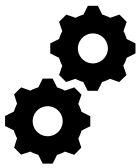
But we never were satisfied with those topics, we always wanted something special for our graduation project, While I'm watching a livestream on Twitch about the new CCNA 200-301, the new section **network automation** got my attention

---

we started looking for it on YouTube after few videos we decided **network automation** is our graduation project

---

Since the delivery duration of the project was very sufficient, we didn't press ourselves with a specific schedule, and we used to work on it whenever the opportunity permitted. After 20 days we finished the project completely.



Working Program

our project is a mixture of networking and programming, we can cover the networking part, but at the time, our had my python knowledge equal to 0

To face this obstacle, I purchased a course from Udemy:

https://www.udemy.com/course/python-network-programming-for-network-engineers-python-3/

we also asked for help from some experienced friends:

BOUZIANE IMANE & ELHOUARI ILIASS

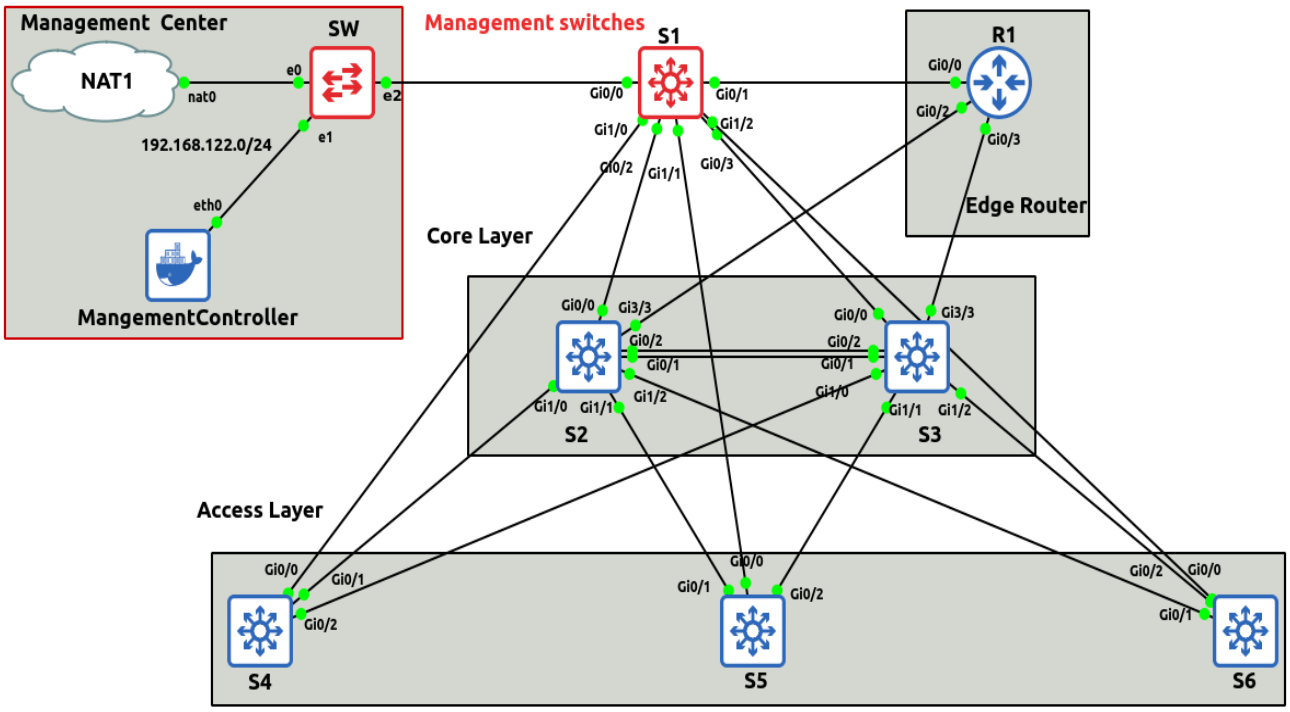
The lesson helped us to organize our thoughts and to draw a vision for the project, once we finished the course, we were able to start

After meeting all the above needs, YASSINE made a list of tools to get started:

- **GNS3 (I had the choice to pick another app but GNS3 was free)**
- **GNS3 VM (a VM to run Qumu devices in GNS3 "*compatible with the hypervisor*")**
- **Cisco IOSV L2 (Switch)**
- **Cisco IOSV L3 (Router)**
- **Network Automation Container (build on Link "Ubuntu")**
- **Python3**
- **Python3 libraries (Netmiko, Paramiko, time, date….)**

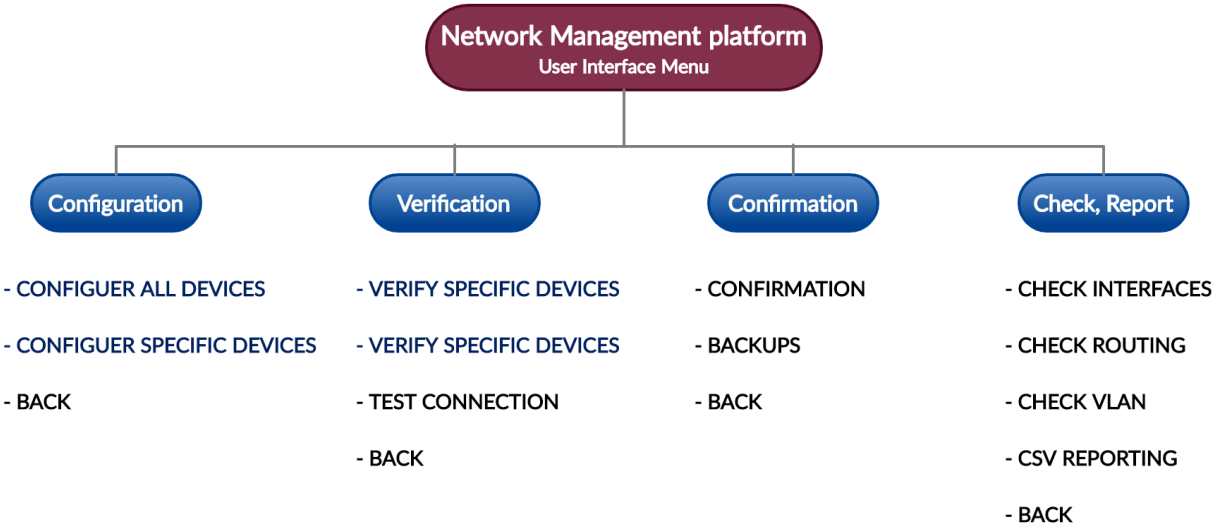
Our project goal is to automate any network, so it doesn't matter what topology you're working on as long it's Made up of Cisco devices

After setting up our topology we started coding



we used this topology but as I said before any Cisco topology will work

we used The Nat cloud only to have internet access to install some requirements



This picture was the vision I had in mind before getting stated and I draw it to work on it

Program organization

Network Automation System

is an application created by Nouhaila & YASSINE , is a Python application for network automation based on Netmiko Library, it is designed to address and facilitate the network management tasks, where we had to put the strategy and make the example for automated process. I cover four common main aspects where automation makes sense, which are:
* Configuration management and provisioning.
* Verification and testing.
* Confirmation and backups.
* Continuous checking and reports generation.

That's all-pretty cover most of an administrator daily work, and they all about pushing and gathering specific information from devices CLI using SSH protocol.

• Libraries:

All used Libraries in the program

```
5  ##########################################################################################################
6                                          #Libraries
7  ##########################################################################################################
8  from netmiko import ConnectHandler
9  from netmiko.ssh_exception import NetMikoTimeoutException, AuthenticationException
10 from paramiko.ssh_exception import SSHException
11 import getpass
12 from time import time, sleep
13 from datetime import datetime
14 import os
15 from simple_term_menu import TerminalMenu
16 from colorama import Fore, Back, Style
17 import csv
18 from threading import Thread
19
20
```

1-Netmiko: is an open-source library designed to simplify SSH management across a wide range of network devices from various vendors including Cisco, Arista, and Juniper Networks.

2-Paramiko: is a Python (2.7, 3.4+) implementation of the SSHv2 protocol, providing both client and server functionality.

3-Simple_term_menu: is a Python implementation creates simple menus for interactive command line programs. It can be used to offer a choice of different options to the user.

4-Threading: is used for creating, controlling, and managing threads in python.

5-Colorama: is a Python implementation Provides a simple cross-platform API to print colored terminal text from **Python** applications

Before executing the code make sure that all libraries are pre-installed in your laptop

Use the commands:

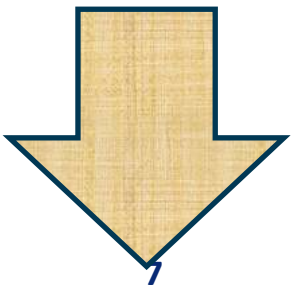
```
# apt-get install python3 pip
```

```
# pip3 install -U "library name"
```

NOTE: apt-get is an **ubuntu** command, use yum in stand if you are in a **redhat** distribution

• functions:

Our code includes 13 functions, whatever compiler you are using type def in the search bare and you will get them easily

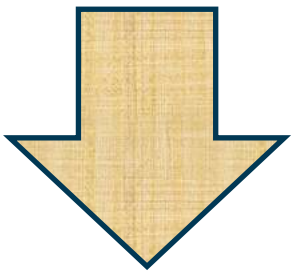


```
1   def ssh_connection(device):
2   def check_version(connection):
3   def device_input():
4   def configuration(device):
5   def verification(device):
6   def test_connection(source, destination, connection):
7   def confirmation(device):
8   def backups(device):
9   def check_interfaces(device):
10  def check_routing(device):
11  def check_vlan(device):
12  def reporting():
13  def main():
```

⇒ ssh_connection(device):

This function handles the ssh connection between the program and cisco device with some instructions.

```
19  ##########################################################################################
20                                      #SSH CONNECTION
21  ##########################################################################################
22
23  def ssh_connection(device):
24          info_device = {
25          'device_type':'cisco_ios',
26          'ip':device[0],
27          'host':device[1],
28          'username': 'nouhaila',
29          'password': 'T@ngerusa98'
30          }
31          try:
32                  connection = ConnectHandler(**info_device)
33          except (AuthenticationException):
34                  print('Authentication failure: '+ ip)
35          except (NetMikoTimeoutException):
36                  print('Timeout to device: ' + ip)
37          except (EOFError):
38                  print('End of file while attempting device: '+ ip)
39          except (SSHException):
40                  print('Be sure that SSH is enabled in: '+ ip +'?')
41          except Exception as unknown_error:
42                  print ('Some other error: '+unknown_error)
43          return connection
44
```



⇒ check_version(device):

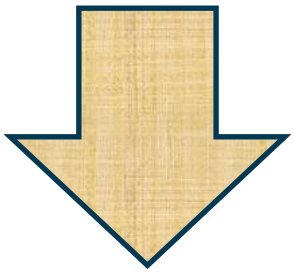
This function Lets you know the device type Router/Switch

```
47  ##############################################################################################
48                                      #CHECK VERSION
49  ##############################################################################################
50
51  def check_version(connection):
52          list_versions = ['vios_l2-ADVENTERPRISEK9-M', 'VIOS-ADVENTERPRISEK9-M']
53          output_version = connection.send_command('show version')
54          for software_ver in list_versions:
55                  int_version = 0
56                  int_version = output_version.find(software_ver)
57                  if int_version > 0:
58                          break
59                  else:
60                          pass
61          return software_ver
62
```

⇒ device_input(device):

This function give you the ability to choose specific devices by asking “do you want more?”

```
69  ##############################################################################################
70                                      #device_input
71  ##############################################################################################
72  def device_input():
73          input_list = []
74          devices_list = []
75          while True:
76                  ip = input(Back.GREEN +'\nEnter the IP address of the device: '+Style.RESET_ALL)
77                  name = input(Back.GREEN +'Enter the hostname : '+Style.RESET_ALL)
78                  input_list.append(ip)
79                  input_list.append(name)
80                  ask = input("\n Do you want more devices? answer by 'y' or 'n'! : " )
81                  devices_list.append(input_list)
82                  input_list = []
83                  if ask == 'y':
84                          continue
85                  elif ask == 'n':
86                          break
87                  else:
88                          input("\n Do you want more devices? answer by 'y' or 'n'! : " )
89          return devices_list
90
```



⇒ configuration(device):

This function allows you to configure your switches and router with one click using 2 files in your PC as a reference: ” router_config_file” & ” switch_config_file”

```
93  ##########################################################################################################
94                                      #CONFIG ALL DEVICES
95  ##########################################################################################################
96  def configuration(device):
97          ip = device [0]
98          name = device [1]
99          connection = ssh_connection(device)
100         print (Back.GREEN+'\nconnection to '+ name + ' is up' + Style.RESET_ALL)
101         software_ver = check_version(connection)
102         if software_ver == 'vios_l2-ADVENTERPRISEK9-M':
103                 print ('Running Switch config file ...')
104                 output = connection.send_config_set(switch_config_file)
105                 print(output)
106         elif software_ver == 'VIOS-ADVENTERPRISEK9-M':
107                 print ('Running Router config_file ...')
108                 output = connection.send_config_set(router_config_file)
109         connection.disconnect()
110
111         return output
112
```

⇒ verification(device):

This function asks the user to enter the adherent name which one to search.

```
114 ##############################################################################################
115                                     #VERIFY ALL DEVICES
116 ##############################################################################################
117 def verification(device):
118         ip = device [0]
119         name = device [1]
120         print(Back.GREEN +'\n'+80*'#'+ Style.RESET_ALL)
121         connection = ssh_connection(device)
122         print (Back.GREEN+'\nconnection to  %s ' %name + ' is up'+ Style.RESET_ALL)
123         software_ver = check_version(connection)
124         if software_ver == 'vios_l2-ADVENTERPRISEK9-M':
125                 running_config = connection.send_command('show running-config')
126                 length = len(switch_config_file)
127                 count = 0
128                 for item in switch_config_file:
129                         if item in running_config:
130                                 count = count + 1
131                         else:
132                                 print(Fore.RED+'{'+ item + '} not found in running-config}'+Style.RESET_ALL)
133                         continue
134                 if count == length:
135                         print(Back.GREEN+'\nCONFIGURATION CORRECT'+Style.RESET_ALL)
136                 else:
137                         print(Back.RED+'\nCONFIGURATION NOT CORRECT'+Style.RESET_ALL)
138         elif software_ver == 'VIOS-ADVENTERPRISEK9-M':
139                 running_config = connection.send_command('show running-config')
140                 length = len(router_config_file)
141                 count = 0
142                 for item in router_config_file:
143                         if item in running_config:
144                                 count = count + 1
145                         else:
146                                 print(Fore.RED+'{'+ item + '} not found in running-config}'+Style.RESET_ALL)
147                         continue
148                 if count == length:
149                         print(Back.GREEN+'\nCONFIGURATION CORRECT'+Style.RESET_ALL)
150                 else:
151                         print(Back.RED+'\nCONFIGURATION NOT CORRECT'+Style.RESET_ALL)
152
```

⇒ test_connection (source, destination, connection):

This function allows to test connectivity between devices by pinging from any source to any destination in the same time

```
156  ##############################################################################################
157                                  #TESTING CONNECTION
158  ##############################################################################################
159  def test_connection(source, destination, connection):
160      ip_source = source [0]
161      name_source = source [1]
162      ip_destination = destination[0]
163      name_destination = destination[1]
164      command = 'ping '+ ip_destination
165      output_ping = connection.send_command(command) #delay_factor = 1)
166      check_list = ['Success rate is 80 percent (4/5)','Success rate is 100 percent (5/5)']
167      if any (item in output_ping for item in check_list):
168          print (Back.GREEN+'Connection from '+name_source+' to '+name_destination+' is reachable==> Success rate'+Style.RESET_ALL)
169      else:
170          print (Back.RED+'Connection from '+name_source+' to '+name_destination
171                 +' is unreachable ==> Check Interfaces and protocols !'+Style.RESET_ALL)
172
```

⇒ confirmation(device):

This function saves the running config of all devices in case you need theme for backup

```
177  ##############################################################################################
178                                      #CONFIRMATION
179  ##############################################################################################
180  def confirmation(device):
181          ip = device [0]
182          name = device [1]
183          connection = ssh_connection(device)
184          print (Back.GREEN+'\nconnection to  %s ' %name +' is up'+ Style.RESET_ALL)
185          saving = connection.save_config()
186          print(saving +'\n---------------------- Succesful-- Saving-------------------------')
187          connection.disconnect()
188
```

⇒ backups(device):

This function allows to edit the books info.

```
193  ##############################################################################################
194                                      #BUCKUPS
195  ##############################################################################################
196  def backups(device):
197          ip = device [0]
198          name = device [1]
199          connection = ssh_connection(device)
200          Backup = connection.send_command("show running-config")
201          file = open("%s_backup.txt" %name ,"w")
202          file.write(Backup)
203          file.close()
204          print(Back.GREEN+"\nBackup for %s is done" %name + Style.RESET_ALL)
205          connection.disconnect()
206
```

⇒ check_interfaces(device):

This function shows the list of all network interfaces in a table which make it easy to analyze interfaces in the network

```
209  ##############################################################################################
210                                      #CHECK INTERFACE
211  ##############################################################################################
212  def check_interfaces(device):
213      ip = device[0]
214      name = device[1]
215      connection=ssh_connection(device)
216      print (Back.GREEN+'\nconnection to  %s ' %name +' is up' +Style.RESET_ALL)
217      output_one = connection.send_command('show int', use_textfsm=True)
218      output_two = connection.send_command('show ip int br', use_textfsm=True)
219      output_three = connection.send_command('show version', use_textfsm=True)
220      i = 0
221      table_data = [[device[1]+'=>'+output_three[0]['version'], 'Interface', 'IP-Address', 'Protocol', 'Status', 'Uptime',
222      'In Error', 'In Pckts', 'Out Error', 'Out Pckts']]
223      while i < len(output_one):
224          int_info = [device[1], output_two[i]['intf'], output_two[i]['ipaddr'],
225          output_two[i]['proto'], output_two[i]['status'], output_three[0]['uptime'],
226          output_one[i]['input_errors'], output_one[i]['input_packets'],
227          output_one[i]['output_errors'], output_one[i]['output_packets']]
228          int_info[1] = int_info[1].replace('GigabitEthernet', 'Gi')
229          if int_info[4] == 'administratively down':
230              int_info[4] = int_info[4].replace('administratively down', 'ad-down')
231          table_data.append(int_info)
232          i = i + 1
233      print(tabulate(table_data, headers="firstrow", tablefmt="fancy_grid", stralign="center", numalign="center"))
234      return table_data
235
```

⇒ check_routing(device):

This function shows routing table which make it easy to analyze routing in the network

```
238  ##############################################################################################
239                                      #CHECK ROUTING
240  ##############################################################################################
241  def check_routing(device):
242          ip = device[0]
243          name = device[1]
244          connection=ssh_connection(device)
245          software_ver = check_version(connection)
246          print (Back.GREEN+'\nconnection to  %s ' %name +' is up'+ Style.RESET_ALL)
247          output_one =connection.send_command('show ip route', use_textfsm=True)
248          output_two =connection.send_command('sh ip ospf database', use_textfsm=True)
249          output_three =connection.send_command('sh ip ospf neighbor', use_textfsm=True)
250          if len(output_three) < 1:
251                  output_three = [{'address':'No neighbor'}]
252          i = 0
253          table_one = [['Ospf\nR-table=>'+device[1], 'Network', 'Mask', 'Next-Hop', 'Protocol', 'Neighbor']]
254          while i < len(output_one):
255                  tt = ['', output_one[i]['network'], output_one[i]['mask'], output_one[i]['nexthop_if'],
256                  output_one[i]['protocol'], output_three[0]['address']]
257                  table_one.append(tt)
258                  i = i + 1
259          print(tabulate(table_one, headers="firstrow", stralign="center", numalign="center", tablefmt="fancy_grid"))
260          table_two = [['Ospf\nData-base=>'+device[1], 'adv_router', 'age', 'area', 'link_count', 'link_id', 'process_id', 'router_id']]
261          if len(output_two) > 0:
262                  j = 0
263                  while j < len(output_two):
264                          tt = ['', output_two[j]['adv_router'], output_two[j]['age'], output_two[j]['area'],
265                          output_two[j]['link_count'], output_two[j]['link_id'], output_two[j]['process_id'], output_two[j]['router_id']]
266                          table_two.append(tt)
267                          j = j + 1
268          print(tabulate(table_two, headers="firstrow", stralign="center", numalign="center", tablefmt="fancy_grid"))
269          return table_one, table_two
```

⇒ check_vlan(device):

This function shows a vlan table which make it easy to analyze Vlan's in the network

```
270  ##############################################################################################
271                                      #CHECK VLAN
272  ##############################################################################################
273  def check_vlan(device):
274      ip = device[0]
275      name = device[1]
276      connection=ssh_connection(device)
277      software_ver = check_version(connection)
278      if software_ver == 'vios_l2-ADVENTERPRISEK9-M':
279          print (Back.GREEN+'\nconnection to  %s ' %name +' is up'+ Style.RESET_ALL)
280          output_one =connection.send_command('show vlan', use_textfsm=True)
281          output_two =connection.send_command('show vtp status', use_textfsm=True)
282          i = 0
283          vlan_data = [[device[1], 'Interfaces', 'Name', 'Status', 'Vlan id', 'VTP-Mode']]
284          while i < len(output_one):
285                  int_info = ['', output_one[i]['interfaces'], output_one[i]['name'],
286                  output_one[i]['status'], output_one[i]['vlan_id'], output_two[0]['mode']]
287                  int_info[1] = ','.join(int_info[1])
288                  vlan_data.append(int_info)
289                  i = i + 1
290          print(tabulate(vlan_data, headers="firstrow", stralign="center", numalign="center", tablefmt="fancy_grid"))
291      else:
292          vlan_data = [['this option can not'], ['be reported']]
293      return vlan_data
```

⇒ reporting():

This function creates a csv file in your pc ,that include all what you need to analyze your network in a csv format

```
294  ##############################################################################################
295                                      #CSV REPORTING
296  ##############################################################################################
297  def reporting():
298          with open('Global_report.csv', 'w', newline='') as file:
299                  writer = csv.writer(file)
300                  writer.writerow(['CHECKING NETWORK IN: ',str(datetime.now())])
301                  writer.writerow([''])
302                  print('\nGenerating a global report of the network ...')
303                  for device in devices_list:
304                          name = device[1]
305                          print (Back.GREEN+'\nconnection to  %s for reporting' %name + Style.RESET_ALL)
306                          writer.writerow([''])
307                          writer.writerow(['THIS IS A SPREADSHEET', 'TO GET INTERFACES INFOS FROM %s' %name])
308                          writer.writerow([''])
309                          writer.writerows(check_interfaces(device))
310                          rout_info= check_routing(device)
311                          writer.writerow([''])
312                          writer.writerow(['THIS IS A SPREADSHEET', 'TO GET ROUTING INFOS FROM %s' %name])
313                          writer.writerow([''])
314                          writer.writerows(rout_info[0])
315                          writer.writerow([''])
316                          writer.writerows(rout_info[1])
317                          writer.writerow([''])
318                          writer.writerow(['THIS IS A SPREADSHEET', 'TO GET VLAN INFOS FROM %s' %name])
319                          writer.writerow([''])
320                          writer.writerows(check_vlan(device))
321                          writer.writerow([''])
322          print('Reporting Done')
```

⇒ main():

This function is a table who combine all the above functions In one table *(this function contains more than 200 lines)*

```
325  ##############################################################################
326                                #UI MENU
327  ##############################################################################
328  def main():
329      main_menu_title = 12 *'*'+Back.CYAN+" 'WELCOME TO THE NETWORK MANAGEMENT PLATFORM MAIN MENU' "+Style.RESET_ALL+ 12 *'*' +"\n"
330      main_menu_items = ["NETWORK CONFIGURATION", "NETWORK VERIFICATION","NETWORK CONFIRMATION", "CHECKING NETWORK", "QUIT"]
331      main_menu_exit = False
332      main_menu = TerminalMenu(menu_entries=main_menu_items, title=main_menu_title)
333
334      conf_menu_title = 24*'*'+Back.CYAN+"'NETWORK CONFIGURATION SECTION'"+Style.RESET_ALL+24*'*'+"\n"
335      conf_menu_items = ["CONFIG ALL DEVICES", "CONFIG SPECIFIC DEVICES", "BACK TO MAIN MENU"]
336      conf_menu_back = False
337      conf_menu = TerminalMenu(conf_menu_items, title=conf_menu_title)
338
339      ver_menu_title = 24*'*'+Back.CYAN+"'NETWORK VERIFICATION SECTION'"+Style.RESET_ALL+24*'*'+"\n"
340      ver_menu_items = ["VERIFY All DEVICES","VERIFY SPECIFIC DEVICES", "TEST CONNECTION", "BACK TO MAIN MENU"]
341      ver_menu_back = False
342      ver_menu = TerminalMenu(ver_menu_items, title=ver_menu_title)
343
344      com_menu_title = 24*'*'+Back.CYAN+"'NETWORK CONFIRMATION SECTION'"+Style.RESET_ALL+24*'*'+"\n"
345      com_menu_items = ["CONFIRMATION", "BUCKUPS", "BACK TO MAIN MENU"]
346      com_menu_back = False
347      com_menu = TerminalMenu(com_menu_items, title=com_menu_title)
348
349      ch_menu_title = 24*'*'+Back.CYAN+"'CHECK NETWORK SECTION'"+Style.RESET_ALL+24*'*'+"\n"
350      ch_menu_items = ["CHECK INTERFACES", "CHECK ROUTING", "CHECK VLAN", "CSV REPORTING","BACK TO MAIN MENU"]
351      ch_menu_back = False
352      ch_menu = TerminalMenu(ch_menu_items, title=ch_menu_title)
353      while not main_menu_exit: ...
463
464
```

More lines here

⇒ MAIN PROGRAMME

This a password that allows the administrator to run the Program.

```
465  ##############################################################################################################
466                                        #MAIN PROGRAMME
467  ##############################################################################################################
468
469  if __name__ == "__main__":
470          startTime = time()
471          with open('switch_config_file') as f:
472                  switch_config_file = f.read().splitlines()
473          with open('router_config_file') as f:
474                  router_config_file = f.read().splitlines()
475          with open('devices_file', 'r') as f:
476                  reader = csv.reader(f)
477                  devices_list = [d for d in reader]
478
479          while True:
480                  username = getpass.getpass(prompt='Username: ')
481                  password = getpass.getpass(prompt='Password: ')
482                  if username.lower() == 'nouhaila'and password =='T@ngerusa98':
483                          break
484                  else:
485                          print('The answer entered by you is incorrect..!!!')
486          main()
```

Qualitative assessment of the work

- **Mechanism of Action:**

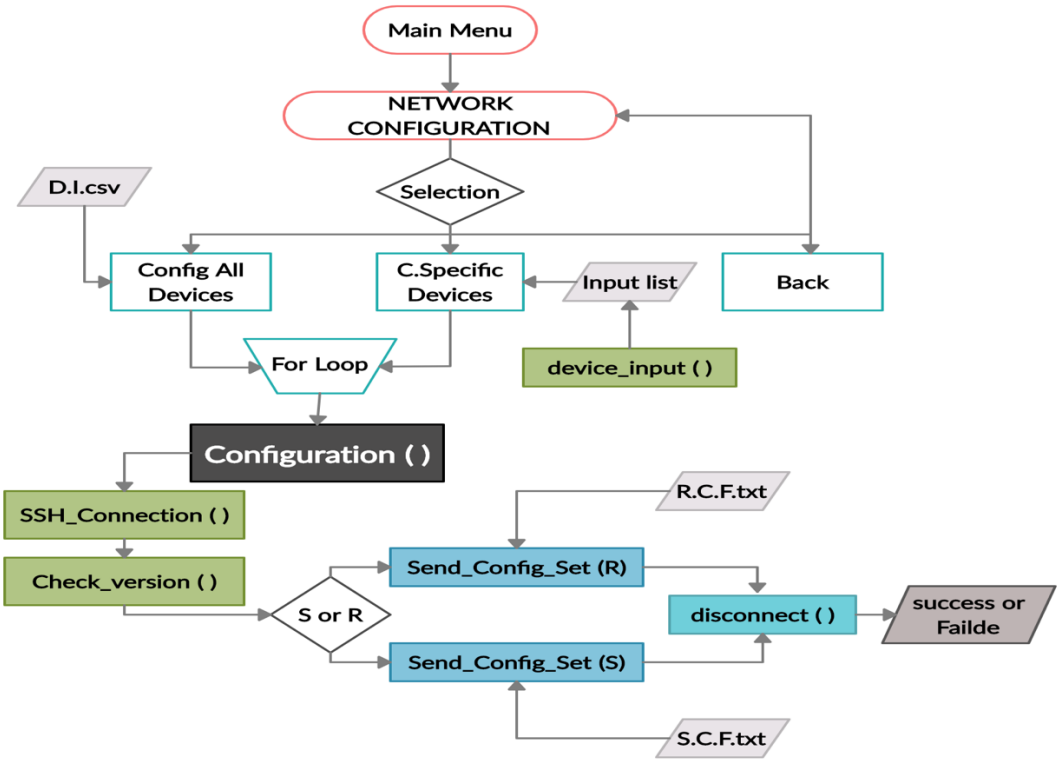
This is the Main Menu Platform with 4 principal button's + the quit button

```
********** 'WELCOME TO THE NETWORK MANAGEMENT PLATFORM MAIN MENU' **

> NETWORK CONFIGURATION
  NETWORK VERIFICATION
  NETWORK CONFIRMATION
  CHECKING NETWORK
  QUIT
```

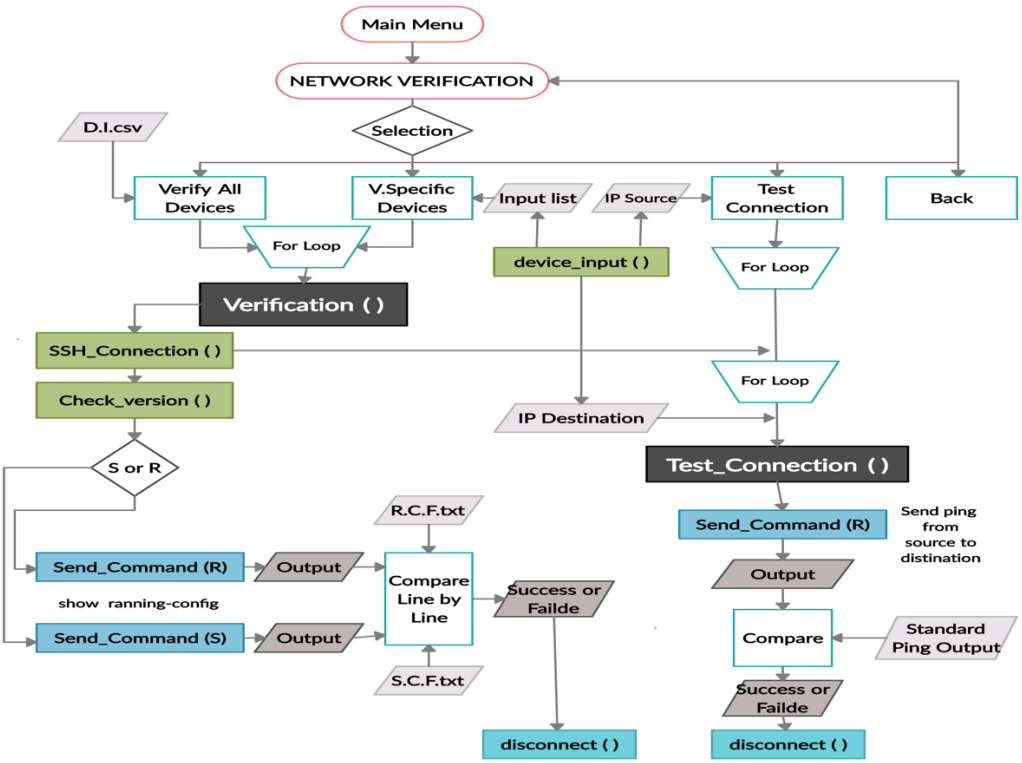
1. Network configuration

A Diagram explain the mechanism of the first button:



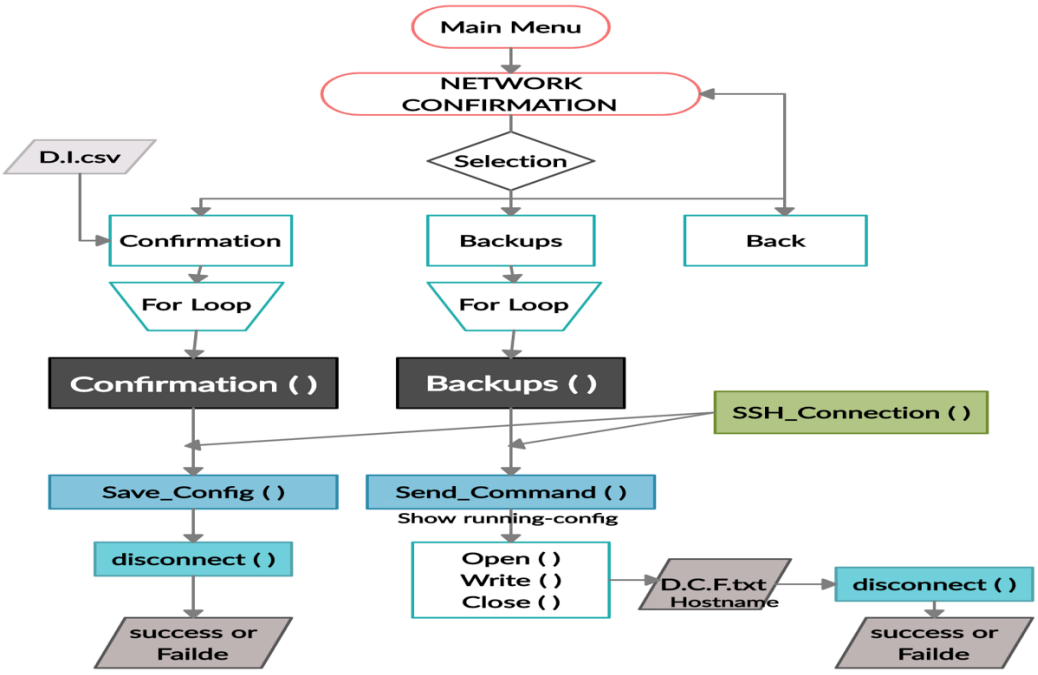
2. Network verification

A Diagram explain the mechanism of the second button:



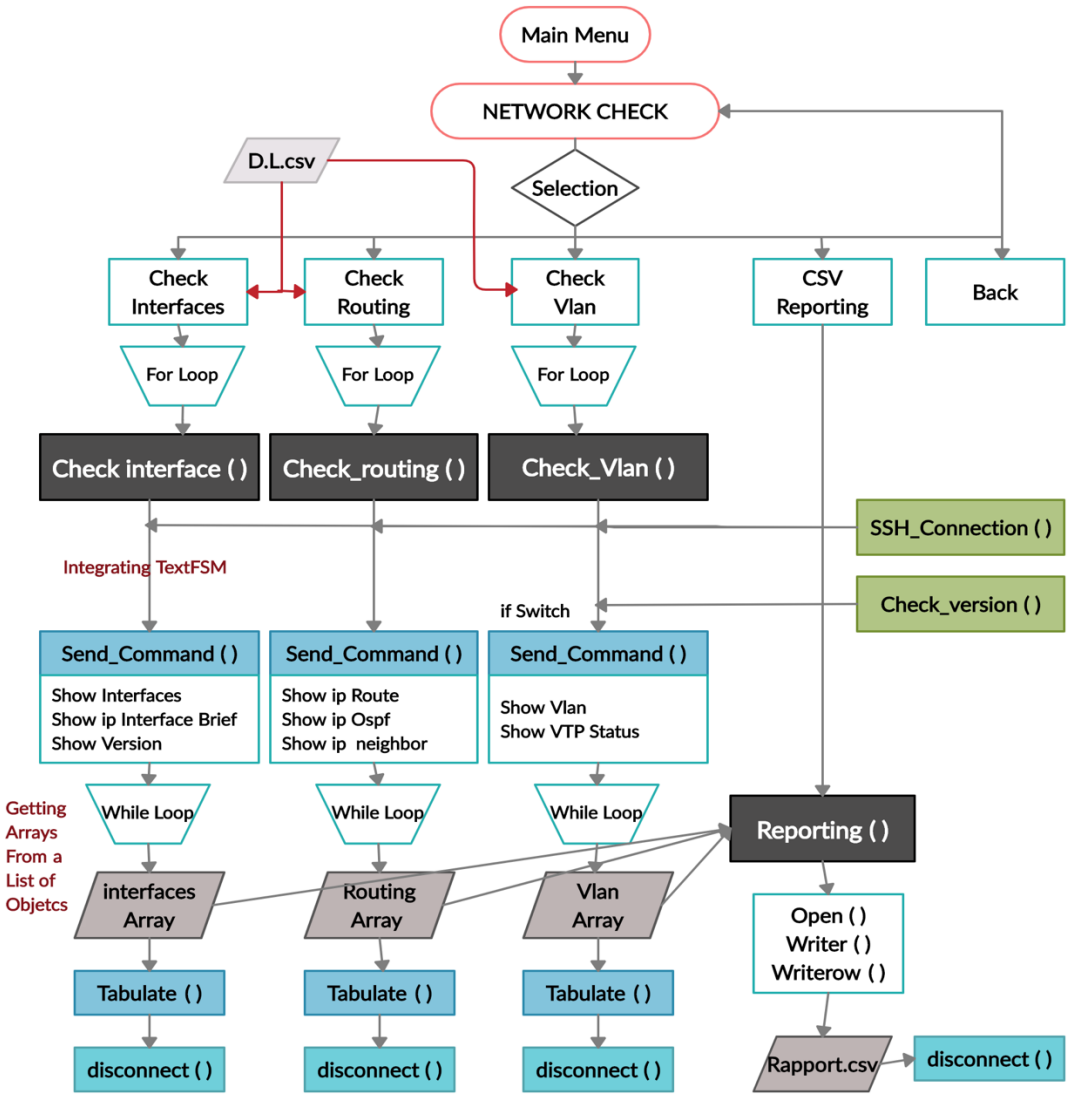
3. Network confirmation

A Diagram explain the mechanism of the third button:



4. Checking network

A Diagram explain the mechanism of the fourth button:



Those diagrams explain the mechanism of the program including all the protocols and the files needed for every operation

- **Difficulties:**

the work on a project we picked was fun, but I can't deny some difficulties that kept us moving slowly:

⇒ **GNS3 ISSUES:**

GNS3 is used by hundreds of thousands of network engineers worldwide to emulate, configure, test, and troubleshoot virtual and real networks. but there are tons of bugs on it, like compatibility of version, compatibility of devices, GNS3-VM issues ...

⇒ **lack of resources:**

Network automation is a new hack in the network world, its hard to find a satisfying explication, specially on the libraries we used for the program, some of them are on beta version

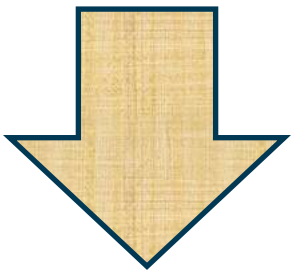
⇒ **Not going in the same direction:**

At some point of the project, I disagreed a lot about how to formulate some functions, with my Teammate which led us to waste a lot of time and effort.

⇒ **Debugging**

It was the worst part of the project, after working for days I was really exited to run our program for the first time and test all the features we worked on, a long list of bugs appears on my screen we had to correct them all

Even those difficulties, we were verry satisfied with the results



- **Screenshots:**

**********'CHECK NETWORK SECTION'**********

Check interfaces Has Been Selected

connection to R1 is up

R1=>15.6(2)T	Interface	IP-Address	Protocol	Status	Uptime	In Error	In Pckts	Out Error	Out Pckts
R1	Gi0/0	192.168.122.211	up	up	2 hours, 28 minutes	0	6000	0	6366
R1	Gi0/1	192.168.1.42	down	down	2 hours, 28 minutes	0	0	0	0
R1	Gi0/2	10.10.10.2	up	up	2 hours, 28 minutes	0	6	0	513
R1	Gi0/3	11.10.10.2	up	up	2 hours, 28 minutes	0	6	0	513
R1	NVI0	192.168.122.211	up	up	2 hours, 28 minutes	0	0	0	0

Check Routing Has Been Selected

connection to R1 is up

Ospf R-table=>R1	Network	Mask	Next-Hop	Protocol	Neighbor
	10.0.0.0	8	GigabitEthernet0/2	C	No neighbor
	10.10.10.2	32	GigabitEthernet0/2	L	No neighbor
	11.0.0.0	8	GigabitEthernet0/3	C	No neighbor
	11.10.10.2	32	GigabitEthernet0/3	L	No neighbor
	192.168.122.0	24	GigabitEthernet0/0	C	No neighbor
	192.168.122.211	32	GigabitEthernet0/0	L	No neighbor

Ospf Data-base=>R1	adv_router	age	area	link_count	link_id	process_id	router_id
	192.168.122.211	1521	0	2	192.168.122.211	100	192.168.122.211

connection to S2 is up

connection to S6 is up

S6	Interfaces	Name	Status	Vlan id	VTP-Mode
	Gi0/0,Gi0/3,Gi1/0,Gi1/1,Gi1/2,Gi1/3,Gi2/0,Gi2/1,Gi3/0,Gi3/1,Gi3/2,Gi3/3	default	active	1	Transparent
	Gi2/2,Gi2/3	Data	active	100	Transparent
	Gi2/2,Gi2/3	Voice	active	101	Transparent
		Test	active	102	Transparent
		fddi-default	act/unsup	1002	Transparent
		token-ring-default	act/unsup	1003	Transparent
		fddinet-default	act/unsup	1004	Transparent
		trnet-default	act/unsup	1005	Transparent

File Edit View Search Terminal Help

**********'NETWORK CONFIRMATION SECTION'**********

Confirm All Devices Has Been Selected

connection to R1 is up
write mem
Building configuration...

[OK]
R1#
---------- Succesful-- Saving----------

connection to S2 is up
write mem
Building configuration...

Compressed configuration from 4295 bytes to 1952 bytes[OK]
S2#
---------- Succesful-- Saving----------

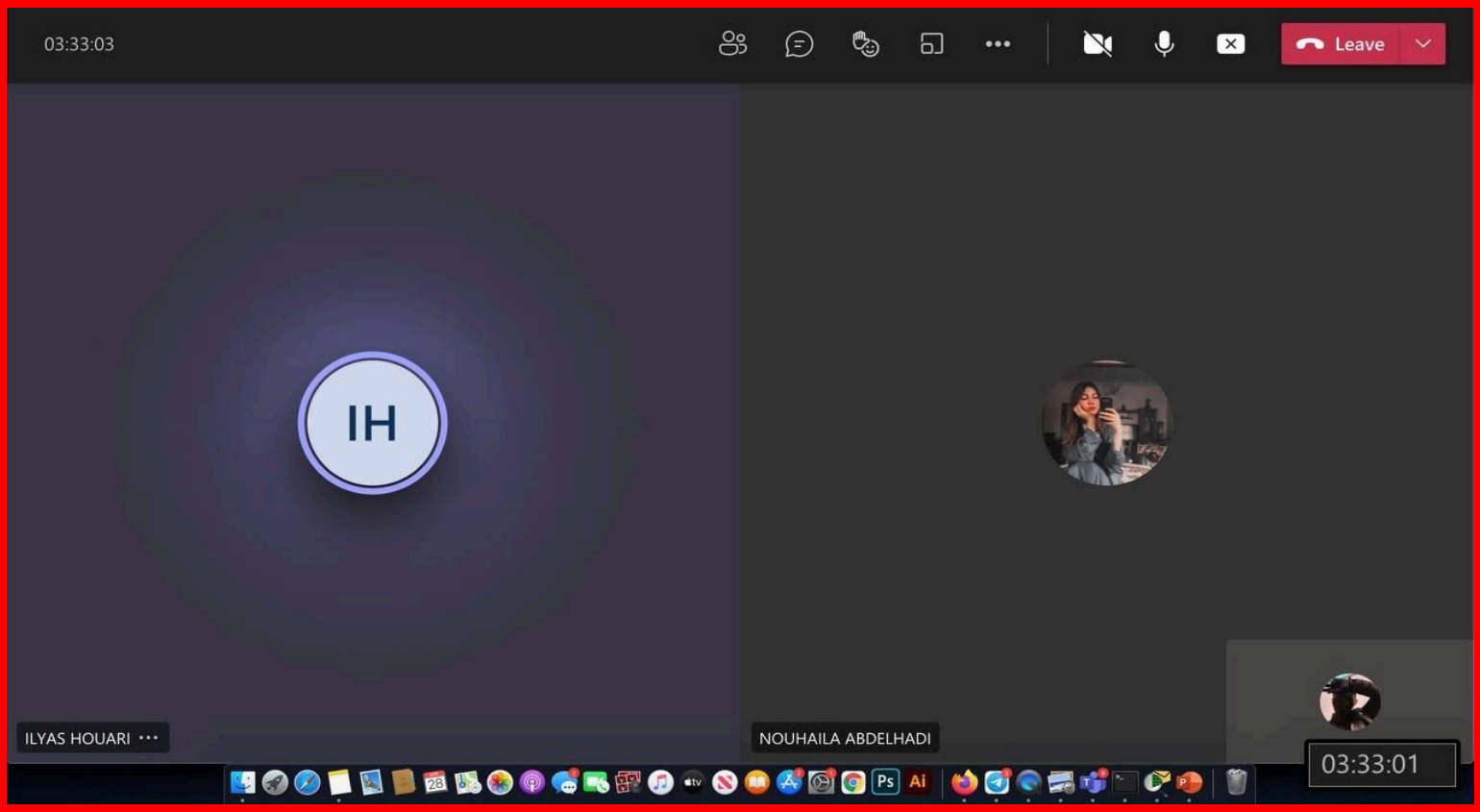
Connection to R1

Connection from R1 to S4 is reachable==> Success rate
Connection from R1 to S5 is reachable==> Success rate
Connection from R1 to S10 is unreachable ==> Check Interfaces and protocols !

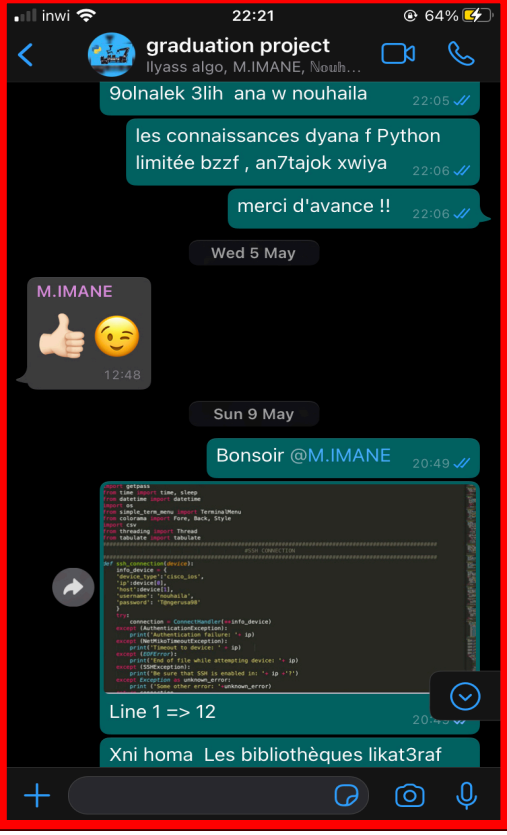
Connection to S1

Connection from S1 to S4 is reachable==> Success rate
Connection from S1 to S5 is reachable==> Success rate
Connection from S1 to S10 is unreachable ==> Check Interfaces and protocols !

Team Calls:


03:33:03
Leave
IH
ILYAS HOUARI
NOUHAILA ABDELHADI
03:33:01


22:07
56%
Back
Nouhaila
00:25
camera
mute
speaker


22:21
64%
graduation project
Ilyass algo, M.IMANE, Nouh...
9olnalek 3lih ana w nouhaila 22:05
les connaissances dyana f Python limitée bzzf , an7tajok xwiya 22:06
merci d'avance !! 22:06
Wed 5 May
M.IMANE
12:48
Sun 9 May
Bonsoir @M.IMANE 20:49
Line 1 => 12
Xni homa Les bibliothèques likat3raf


Yassin
01:34
Terminer
Muet
Activer vidéo
Haut-parleur

Conclusion

The purpose of this project was to identify effective strategies for dealing with repetitive network Tasks Based on a python3 code, it can be concluded that network automation is very important for the improvement any network.

Every network engineer should learn to automate networks not just to save time also to simplify the boring usual routine of

Python is not the only way to automate networks, there is hundreds of methods to automate like ansible, napalm, paramiko ...

Finally, a special thanks to my friends ILIASS ELHOUARI and IMANE BOUZIANE, without them, it would never be possible to make this project work