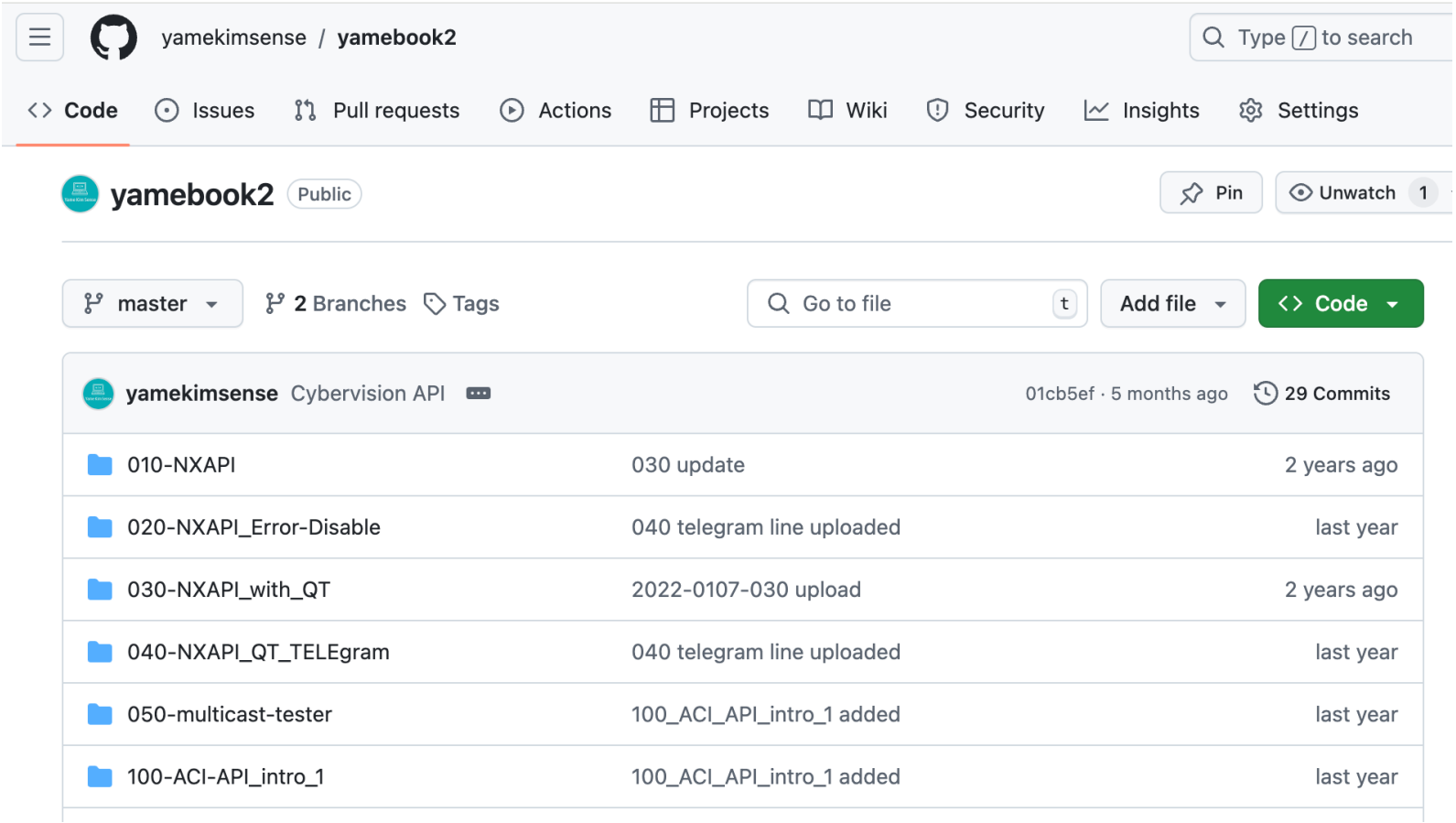
Auto VLAN Assign by IP address for Cisco Catalyst

2024. 11.

Visit Gibhub.com for code and text

https://github.com/yamekimsense/yamebook2



Customer Request

- windows PC IP address change batch file
- Automatically VLAN assigned to Cisco Catalyst port base upon the PC IP address

Topology

Process Flow

0) Preparation - Catalyst netflow configuration

1) PC connection or IP change using **batch (netsh)**

2) Python Code Activation by **EEM** (EEM@IOS-XE CLI)

3) Python Code on **Guestshell**

Port number detection

IP address detection (from netflow)

if data is (\n)

if ip address (.)

getting VLAN from ***modVLANFinder.VLANFinder***

apply VLAN using cli.clip

Windows11 - IPv4 Packet Burst when NIC starting

Multicast packets are sent! ➔ Netflow can detect the source IP.

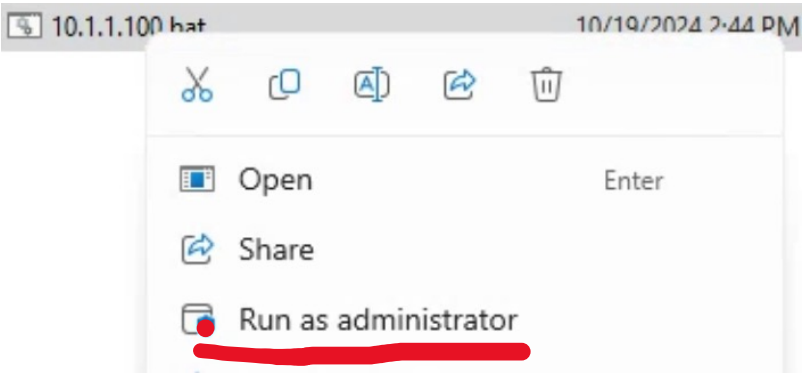
ip

No.	Time	Time	Source	Destination	Protocol	Length	Type	Info
1	2024-10-20 15:52:18.175122	0.000000	10.1.1.100	224.0.0.252	LLMNR	64	IPv4	Standard query 0xa9b6 A wpad
2	2024-10-20 15:52:18.176162	0.001040	10.1.1.100	224.0.0.252	LLMNR	64	IPv4	Standard query 0x9564 AAAA wpad
3	2024-10-20 15:52:18.553719	0.378597	10.1.1.100	224.0.0.251	IGMPv2	46	IPv4	Membership Report group 224.0.0.251
4	2024-10-20 15:52:18.553719	0.378597	10.1.1.100	224.0.0.252	IGMPv2	46	IPv4	Membership Report group 224.0.0.252
5	2024-10-20 15:52:18.555711	0.380589	10.1.1.100	224.0.0.251	MDNS	81	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA.local, "QM" question
6	2024-10-20 15:52:18.558888	0.383766	10.1.1.100	224.0.0.251	MDNS	81	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA.local, "QM" question
7	2024-10-20 15:52:18.559230	0.384108	10.1.1.100	239.255.255.250	IGMPv2	46	IPv4	Membership Report group 239.255.255.250
8	2024-10-20 15:52:18.562229	0.387107	10.1.1.100	224.0.0.251	MDNS	119	IPv4	Standard query response 0x0000 AAAA fe80::38cb:d4e3:c82c:7cc0 A 10
9	2024-10-20 15:52:18.563055	0.387933	10.1.1.100	224.0.0.251	MDNS	119	IPv4	Standard query response 0x0000 AAAA fe80::38cb:d4e3:c82c:7cc0 A 10
10	2024-10-20 15:52:18.593417	0.418295	10.1.1.100	224.0.0.252	LLMNR	64	IPv4	Standard query 0xa9b6 A wpad
11	2024-10-20 15:52:18.593417	0.418295	10.1.1.100	224.0.0.252	LLMNR	64	IPv4	Standard query 0x9564 AAAA wpad
12	2024-10-20 15:52:18.782469	0.607347	10.1.1.100	239.255.255.250	SSDP	179	IPv4	M-SEARCH * HTTP/1.1
13	2024-10-20 15:52:18.800300	0.625178	192.168.101.1	224.0.0.13	PIMv2	72	IPv4	Hello
14	2024-10-20 15:52:18.800300	0.625178	192.168.101.1	224.0.0.13	PIMv2	62	IPv4	Assert
15	2024-10-20 15:52:18.814327	0.639205	10.1.1.100	239.255.255.250	SSDP	179	IPv4	M-SEARCH * HTTP/1.1
16	2024-10-20 15:52:18.893077	0.717955	10.1.1.100	239.255.255.250	SSDP	179	IPv4	M-SEARCH * HTTP/1.1
17	2024-10-20 15:52:19.034203	0.859081	10.1.1.100	224.0.0.251	IGMPv2	46	IPv4	Membership Report group 224.0.0.251
18	2024-10-20 15:52:19.034244	0.859122	10.1.1.100	224.0.0.252	IGMPv2	46	IPv4	Membership Report group 224.0.0.252
19	2024-10-20 15:52:19.034267	0.859145	10.1.1.100	239.255.255.250	IGMPv2	46	IPv4	Membership Report group 239.255.255.250
20	2024-10-20 15:52:19.547837	1.372715	10.1.1.100	224.0.0.251	MDNS	93	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA._dosvc._tcp.local, "QU"
21	2024-10-20 15:52:19.767693	1.592571	10.1.1.100	224.0.0.251	MDNS	93	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA._dosvc._tcp.local, "QU"
22	2024-10-20 15:52:19.866959	1.691837	10.1.1.100	239.255.255.250	SSDP	179	IPv4	M-SEARCH * HTTP/1.1
23	2024-10-20 15:52:19.907443	1.732321	10.1.1.100	224.0.0.251	MDNS	93	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA._dosvc._tcp.local, "QU"
24	2024-10-20 15:52:20.035616	1.860494	10.1.1.100	224.0.0.251	MDNS	344	IPv4	Standard query response 0x0000 SRV, cache flush 0 0 7680 DESKTOP-S
25	2024-10-20 15:52:20.544393	2.369271	10.1.1.100	224.0.0.2	IGMPv2	46	IPv4	Leave Group 224.0.0.251
26	2024-10-20 15:52:20.544827	2.369705	10.1.1.100	224.0.0.2	IGMPv2	46	IPv4	Leave Group 224.0.0.252
27	2024-10-20 15:52:20.564842	2.389720	10.1.1.100	224.0.0.251	IGMPv2	46	IPv4	Membership Report group 224.0.0.251
28	2024-10-20 15:52:20.564924	2.389802	10.1.1.100	224.0.0.252	IGMPv2	46	IPv4	Membership Report group 224.0.0.252
29	2024-10-20 15:52:20.571836	2.396714	10.1.1.100	224.0.0.251	MDNS	81	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA.local, "QM" question
30	2024-10-20 15:52:20.573751	2.398629	10.1.1.100	224.0.0.251	MDNS	119	IPv4	Standard query response 0x0000 AAAA fe80::38cb:d4e3:c82c:7cc0 A 10
31	2024-10-20 15:52:20.573918	2.398796	10.1.1.100	224.0.0.251	MDNS	81	IPv4	Standard query 0x0000 ANY DESKTOP-SL033KA.local, "QM" question
32	2024-10-20 15:52:20.575509	2.400387	10.1.1.100	224.0.0.251	MDNS	119	IPv4	Standard query response 0x0000 AAAA fe80::38cb:d4e3:c82c:7cc0 A 10
33	2024-10-20 15:52:20.586463	2.411341	10.1.1.100	224.0.0.251	MDNS	70	IPv4	Standard query 0x0000 A wpad.local, "QM" question
34	2024-10-20 15:52:20.591426	2.416304	10.1.1.100	224.0.0.251	MDNS	70	IPv4	Standard query 0x0000 AAAA wpad.local, "QM" question
35	2024-10-20 15:52:20.593829	2.418707	10.1.1.100	224.0.0.251	MDNS	70	IPv4	Standard query 0x0000 A wpad.local, "QM" question

0) IP address change Batch file

"Ethernet" is NIC name of Windows. Run the batch file as <administrator>.

```
netsh interface ip set address "Ethernet" static 10.1.1.100 255.255.255.0 10.1.1.1
netsh interface set interface "Ethernet" disable
timeout 2
netsh interface set interface "Ethernet" enable
timeout 10
ipconfig
pause
```



https://learn.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-contexts

```
Commands in this context:
..            - Goes up one context level.
?             - Displays a list of commands.
abort         - Discards changes made while in offline mode.
add           - Adds a configuration entry to a list of entries.
advfirewall   - Changes to the `netsh advfirewall' context.
alias         - Adds an alias.
branchcache   - Changes to the `netsh branchcache' context.
bridge        - Changes to the `netsh bridge' context.
bye           - Exits the program.
commit        - Commits changes made while in offline mode.
delete        - Deletes a configuration entry from a list of entries.
dhcpclient    - Changes to the `netsh dhcpclient' context.
dnsclient     - Changes to the `netsh dnsclient' context.
dump          - Displays a configuration script.
exec          - Runs a script file.
exit          - Exits the program.
firewall      - Changes to the `netsh firewall' context.
help          - Displays a list of commands.
http          - Changes to the `netsh http' context.
interface     - Changes to the `netsh interface' context.
ipsec         - Changes to the `netsh ipsec' context.
ipsecdosprotection - Changes to the `netsh ipsecdosprotection' context.
lan           - Changes to the `netsh lan' context.
namespace     - Changes to the `netsh namespace' context.
netio         - Changes to the `netsh netio' context.
offline       - Sets the current mode to offline.
online        - Sets the current mode to online.
popd          - Pops a context from the stack.
pushd         - Pushes current context on stack.
quit          - Exits the program.
ras           - Changes to the `netsh ras' context.
rpc           - Changes to the `netsh rpc' context.
set           - Updates configuration settings.
show          - Displays information.
trace         - Changes to the `netsh trace' context.
unalias       - Deletes an alias.
wfp           - Changes to the `netsh wfp' context.
winhttp       - Changes to the `netsh winhttp' context.
winsock       - Changes to the `netsh winsock' context.

The following sub-contexts are available:
advfirewall branchcache bridge dhcpclient dnsclient firewall http interface ipsec
ipsecdosprotection lan namespace netio ras rpc trace wfp winhttp winsock

To view help for a command, type the command, followed by a space, and then type ?.
```

1) NETFLOW result

PC IP address acqusition From netflow table ***IPV4 SRC ADDR***

```
C9300_17.9.4a#show flow monitor yame-flow cache format table | include 10.
  Cache size:                                 10000
10.1.1.100
```

```
C9300_17.9.4a#show flow monitor yame-flow cache format table
  Cache type:                                 Normal (Platform cache)
  Cache size:                                 10000
  Current entries:                                1

  Flows added:                                    2
  Flows aged:                                     1
    - Inactive timeout    (    15 secs)           1

IPV4 SRC ADDR
===============
10.1.1.100
```

1) NETFLOW configuration

netflow is cisco proprietary

```
flow record yame-flow
 match ipv4 source address
!
flow monitor yame-flow
 record yame-flow
!
interface GigabitEthernet1/0/48
 ip flow monitor yame-flow input
```

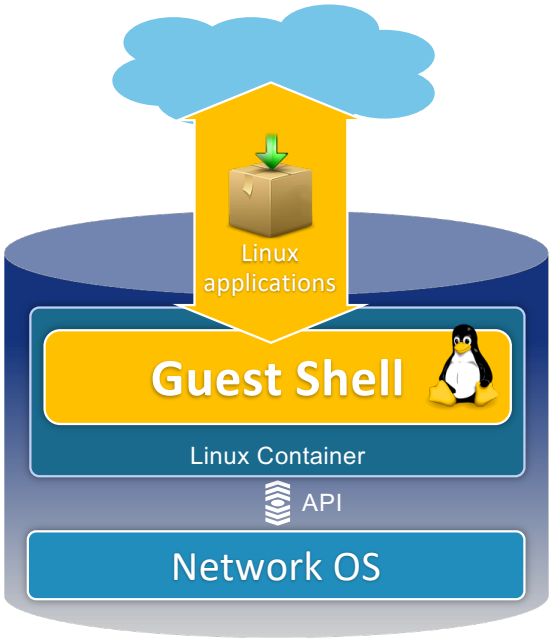
2) EEM (Embedded Event Manager)

EEM detects the port up log and run the python script.

```
event manager applet G1/0/48_up_event
 event syslog pattern "LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/48, changed state to up"
 action 1.0 cli command "send log ##### EEM started for 1/0/48 - Wait 5 sec"
 action 2.0 cli command "enable"
 action 5.0 wait 5
 action 6.0 cli command "send log ##### python code called"
 action 8.0 cli command "guestshell run python3 d610-48.py"
 action 9.0 cli command "send log ##### EEM completed for 1/0/48"
```

3) What is the Guest Shell

- 64-bit Linux environment running on IOS XE and NX-OS platforms
- Install, update, and operate 3rd party Linux apps (e.g. Puppet, Chef, Splunk)
- Bundled with Python
- Intended for agent or script hosting



Cisco Guest Shell Capabilities

	Guest Shell 1.0 (Lite)	Guest Shell 1.0	Guest Shell 2.1
Operating System	IOS-XE 16.5.1a	IOS-XE 16.5	NX-OS 7.x
Platforms	CAT 3650, CAT3850	CAT 9000, ISR 4000	Nexus 3000, 9000
Guest Shell Environment	MontaVista CGE7	CentOS 7	CentOS 7
Python 2.7	✓	✓	✓
Python 3.0	✗	✓	✓
Python GNU C Compiler	✗	✗	✓
RPM Install	✗	✓	✓
OVA Enable/Upgrade	✗	✗	✓
User Defined Resources	✗	✗	✓

<dohost> CLI access from Guestshell

To use <dohost>, shell command os.popen or another command is needed.

example	

`[guestshell@guestshell ~]$ dohost "show clock"`
`14:10:33.850 UTC Sun Oct 20 2024`
 	
`[guestshell@guestshell ~]$`
 	
The **dohost** command requires the **ip http server** command to be configured on the device.	

`import os`
 	
`a = 'dohost \"show clock\"'`
`stream = os.popen(a)`
`output = stream.read()`
`print(type(output))`
`print (output)`
	`[guestshell@guestshell ~]$ python3 dohost.py`
	`<class 'str'>`
	`14:37:25.259 UTC Sun Oct 20 2024`
	
	`[guestshell@guestshell ~]$`

CLI module for python needs <import cli>

example	

cli.cli(command) —This function takes an IOS command as an argument, runs the command through the IOS parser, and ***returns the resulting text***. If this command is malformed, a Python exception is raised. The following is sample output from the **cli.cli(command)** function:	`>>> a = cli.cli('show clock; show clock')`
	`>>> print (type(a))`
	`<class 'str'>`
	`>>> print (a)`
	`14:26:24.606 UTC Sun Oct 20 2024`
	`14:26:24.908 UTC Sun Oct 20 2024`

cli.clip(command) —This function works exactly the same as the **cli.cli(command)** function, except that it prints the resulting text to ***stdout*** rather than returning it. The following is sample output from the **cli.clip(command)** function:	`>>> a = cli.clip('show clock; show clock')`
	`14:25:29.054 UTC Sun Oct 20 2024`
	`14:25:29.355 UTC Sun Oct 20 2024`
	
	`>>> print (type(a))`
	`<class 'NoneType'>`

cli.execute(command) —This function executes a ***single EXEC command*** and returns the output; however, ***does not print the resulting text*** No semicolons or newlines are allowed as part of this command. Use a Python list with a for-loop to execute this function more than once. The following is sample output from the **cli.execute(command)**	`>>> a = cli.execute("show clock")`
	`>>> print (type(a))`
	`<class 'str'>`
	`>>> print (a)`
	`14:27:06.937 UTC Sun Oct 20 2024`
	
	`>>> a = cli.execute("show clock; show clock")`
	`Traceback (most recent call last):`

CLI module for python needs <import cli>

example	

cli.executep(command) –This function executes a single command and ***prints the resulting text to stdout*** rather than returning it. The following is sample output from the **cli.executep(command)** function:	`>>> a= cli.executep('show clock')`
	`14:28:21.263 UTC Sun Oct 20 2024`
	
	`>>> print (type(a))`
	`<class 'NoneType'>`
	`>>> print (a)`
	`None`
	`>>>`
	`>>>`
	`>>> a= cli.executep('show clock; show clock')`
	`You may not run multiple commands using execute().: There was a problem running the command: "show clock; show clock"`
	`>>>`
	`>>> print (type(a))`
	`<class 'NoneType'>`
	`>>> print (a)`
	`None`
	`>>>`

CLI module for python needs <import cli>

	example

cli.configure(command) –This function configures the device with the configuration available in commands. ***It returns a list of named tuples*** that contains the command and its result as shown below:
 	
The command parameters can be in multiple lines and in the same format that is displayed in the output of the **show running-config** command. The following is sample output from the **cli.configure(command)** function:
	`>>> a = cli.configure(["interface GigabitEthernet1/0/7", "shutdown", "end"])`
	`>>> print (type(a))`
	`<class 'str'>`
	`>>> print (a)`
	`Line 1 SUCCESS:  interface GigabitEthernet1/0/7`
	`Line 2 SUCCESS:  shutdown`
	`Line 3 SUCCESS:  end`

cli.configurep(command) –This function works exactly the same as the **cli.configure(command)** function, except that it prints the resulting text to *stdout* rather than returning it. The following is sample output from the **cli.configurep(command)** function:	`>>> a= cli.configurep(["interface GigabitEthernet1/0/7", "no shutdown", "end"])`
	`Line 1 SUCCESS:  interface GigabitEthernet1/0/7`
	`Line 2 SUCCESS:  no shutdown`
	`Line 3 SUCCESS:  end`
	
	`>>> print (type(a))`
	`<class 'NoneType'>`
	`>>> print (a)`
	`None`

3) file list

Use "vi editor" because it's CentOS

```
[guestshell@guestshell ~]$ pwd
/home/guestshell
```

```
[guestshell@guestshell ~]$ ls -l
total 7
-rw-rw-r--. 1 guestshell users   50 Oct 20 15:01 VLAN_info.txt
drwxrwxr-x. 2 guestshell users 1024 Oct 20 15:01 __pycache__
-rw-rw-r--. 1 guestshell users  117 Oct 20 15:11 cli_test.py
-rw-rw-r--. 1 guestshell users 2668 Oct 20 15:29 d610-48.py
-rw-rw-r--. 1 guestshell users  668 Oct 20 15:01 modVLANFinder.py
```

3) Enable Guestshell and IP address

- C9300_17.9.4a(config)#iox
- C9300_17.9.4a#guestshell enable
- Address Configuration

```
interface VirtualPortGroup0
 ip address 192.168.35.1 255.255.255.0

app-hosting appid guestshell
 app-vnic gateway1 virtualportgroup 0 guest-interface 0
  guest-ipaddress 192.168.35.2 netmask 255.255.255.0
 app-default-gateway 192.168.35.1 guest-interface 0
```

4) Python Code : VLAN_info.txt /// modVLANFinder.py

```
100,10.1.1.0/24
200,10.1.2.0/25
201,10.1.2.128/25
```

```python
import ipaddress

def VLANFinder(IP_address):
    f = open("VLAN_info.txt", 'r')
    lines = f.readlines()
    for line in lines:
        #print(line)
        VLAN_ID = line.split(",")[0]
        NETWORK_MASK = line.split(",")[1].replace("\n","")
        #print ("#### line 8", VLAN_ID, NETWORK_MASK)

        if ipaddress.ip_address(IP_address) in ipaddress.ip_network(NETWORK_MASK) :
            print ("#### line 14", VLAN_ID, NETWORK_MASK)
            VLAN_ID_RESULT = VLAN_ID
    f.close()
    return VLAN_ID_RESULT

if __name__ == "__main__":
    IP_address = '10.1.100.2'
    VLAN_ID_RESULT = VLANFinder(IP_address)
    print ("#### line 22", VLAN_ID_RESULT)
```

Main Code1

```
#modVLANFinder.py - to determine VLAN ID
#VLAN_info.txt - vlan id and subnet mask

import os, cli
import modVLANFinder #import VLANFinder.py
import inspect

#from file name, find the port number
#d610-48.py is 48th port i.e 1/0/48
file_name = inspect.getfile(inspect.currentframe())
print (file_name)
file_name = file_name.split('-')
how_many = len(file_name)
print (how_many)
port_number = file_name[how_many - 1]
port_number = port_number.replace(".py","")
print ("port number", port_number)

# from netflow table, find the IP address
a = 'dohost \"show flow monitor yame-flow cache format table | include 10\.\"'
stream = os.popen(a)
output = stream.read()
print("Source IP address", type(output), output)
```

Main Code2

```
if "\n" in output:   #if line is there, run this line. If not, do nothing.
    print("39th line n detected")
    IP_address = output.replace("\n","").replace(" ","").replace(" ","")

    message = "line_is_detected"
    message = f'dohost \"send log \"****{message}\"\"'
    stream = os.popen(message)

    if IP_address.count('.') > 2: #if IP address is there, run thie if. If not, do nothing.
        message = "source_address_is_" + IP_address
        message = f'dohost \"send log \"****{message}\"\"'
        stream = os.popen(message)

        VLAN_ID = modVLANFinder.VLANFinder (IP_address)
        command1 = 'interface g1/0/' + port_number
        command2 = 'switch access vlan ' + VLAN_ID

        message = "VLAN_ID_is_" + VLAN_ID
        message = f'dohost \"send log \"****{message}\"\"'
        stream = os.popen(message)

        print("VLAN_APPLY_command:::", command1, command2)
        result = cli.configure([command1, command2])
        print ("Applied result ", type(result), result)

        message = result.replace("\n","_").replace("\n","_").replace("  ","_").replace("
","_").replace(" ","_").replace(" ","_").replace(" ","_").replace(" ","_").replace(" ","_").r
","_").replace(" ","_").replace(" ","_").replace(" ","_").replace(" ","_").replace(" ","_").r
","_").replace(" ","_").replace(" ","_")
        print ("message is", message)
        message = f'dohost \"send log \"****{message}\"\"'
        stream = os.popen(message)
```

Run Example

```
C9300_17.9.4a#
Oct 20 15:29:56.498: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/48, changed
state to down
Oct 20 15:29:57.497: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/48, changed state to down
Oct 20 15:30:02.111: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/48, changed state to up
Oct 20 15:30:03.110: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/48, changed
state to up
Oct 20 15:30:08.368: %SYS-7-USERLOG_DEBUG: Message from tty3(user id: ): ##### python code called
Oct 20 15:30:14.320: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: shxUnknownTTY):
****source_address_is_10.1.1.100
Oct 20 15:30:14.326: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: shxUnknownTTY):
****line_is_detected
Oct 20 15:30:14.329: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: shxUnknownTTY):
****VLAN_ID_is_100
Oct 20 15:30:15.618: %SYS-7-USERLOG_DEBUG: Message from tty3(user id: ): ##### EEM completed for
1/0/48
Oct 20 15:30:21.462: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: cisco): ****line_is_detected
Oct 20 15:30:21.464: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: cisco): ****VLAN_ID_is_100
Oct 20 15:30:21.464: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: cisco):
****source_address_is_10.1.1.100
Oct 20 15:30:22.689: %SYS-7-USERLOG_DEBUG: Message from tty73(user id: cisco):
****Line_1_SUCCESS:_interface_g1/0/48_Line_2_SUCCESS:_switch_access_vlan_100_
C9300_17.9.4a#
```