Автоматизация 5. Netmiko на практике

Предыдущие статьи цикла:

- Автоматизация работы сетевого администратора
- Автоматизация работы сетевого администратора, часть 2. Практические кейсы с Telnetlib
- Автоматизация работы сетевого администратора. Часть 3. Знакомство с Paramiko
- Автоматизация 4. Netmiko

Всех приветствую, продолжаем осваивать программные инструменты и автоматизировать рутинные процессы сетевого администрирования. Сегодняшняя статья будет в основном про практику, будут следовать примеры и описания работы скриптов, все также изучаем Netmiko и Python (необходимый минимум).

Обработка ошибок

До сих пор мы рассматривали различные решения и не касались работы с ошибками при работе программы со стороны пользователя или сервера (в нашем случае - сетевое оборудование).

Зачастую логика скрипта может быть написана правильно, но он все равно не отработает, предоставив специфический вывод об ошибке. В случае конфигурирования более 1-го сетевого узла, это может привести к выходу из программы и потери конечного ожидаемого результата.

Возможные причины сбоя вашего скрипта:

1. Некорректные аутентификационные данные.
2. Истек таймаут для подключения.
3. Прерывание передачи трафика.
4. Недоступность порта SSH со стороны сервера.
5. Различные другие ошибки.

Согласитесь, было бы удобно, каким-либо образом обрабатывать такие ошибки и сообщать пользователю о конкретной проблеме, а что даже наиболее важно - не прерывать выполнение программы далее.

Для этого в Python есть конструкция `try / except`.

```
try:
  BLOCK
except CODE_ERROR as err:
   print("OS error: {0}".format(err))
```

То есть, в блоке "Try" мы пытаемся выполнить какой-либо код , он может вызвать ошибку, тогда мы пробуем ее обработать и получить о ней вывод - с помощью блока "Except". При этом скрипт продолжит свою работу далее, если это не последние шаги в программе.

Дополнительно есть операторы:

`"Break"` - принудительный выход из цикла, соответственно далее выполняется сама программа.

`"Continue"` - возвращает управление в начало цикла, следовательно позволяет пропустить оставшиеся строки в цикле и перейти к следующей итерации в нем же.

Более подробно о подходе по обработке ошибок в официальном мануале - ССЫЛКА.

Вернемся к нашей библиотеке Netmiko и рассмотрим скрипт, который будет проверять доступность сетевого устройства:

```
#!/usr/bin/env python
"Импорт необходимых библиотек"
from getpass import getpass
from netmiko import ConnectHandler
from netmiko.ssh_exception import NetMikoTimeoutException
from paramiko.ssh_exception import SSHException
from netmiko.ssh_exception import AuthenticationException

"Ввод аутентификационных данных в зашифрованном виде"
username = input("Enter your SSH username: ")
password = getpass()
```

```
"Открытие файла с командами и считывание их в переменную"
with open("commands_file") as f:
    commands_list = f.read().splitlines()

"Открытие файла со списком устройств и считывание их в переменную"
with open("devices_file") as f:
    devices_list = f.read().splitlines()

"Подключение к устройству из заданного списка в цикле FOR"
for devices in devices_list:
    print ("Connecting to device" " + devices)
    ip_address_of_device = devices
    ios_device = {
        "device_type": "cisco_ios",
        "ip": ip_address_of_device,
        "username": username,
        "password": password
    }

"Попытка обработать ошибку"
    try:
        net_connect = ConnectHandler(**ios_device)

    #Срабатывание исключений в случае неудачного создания соединения
    except (AuthenticationException):
        print ("Authentication failure: " + ip_address_of_device)
        continue
    except (NetMikoTimeoutException):
        print ("Timeout to device: " + ip_address_of_device)
        continue
    except (EOFError):
        print ("End of file while attempting device " + ip_address_of_device)
        continue
    except (SSHException):
        print ("SSH Issue. Are you sure SSH is enabled? " + ip_address_of_device)
        continue
    except Exception as unknown_error:
        print ("Some other error: " + str(unknown_error))
        continue
    output = net_connect.send_config_set(commands_list)
    print (output)
```

Таким образом, в случае неудачного SSH-подключения, наш скрипт попробует обработать ошибку согласно заданной логике "except" и вывести на экран информацию. Введем заведомо неправильные значения логин/пароля и выведем на экран результат:

```
>>> try:
...     net_connect = ConnectHandler(**cisco)
... except (AuthenticationException):
...     print ('Authentication failure is really ' + cisco['host'] )
...
Authentication failure is really 192.168.122.72
```

Для удобства показали как скрипт отработает в интерпретаторе с информацией о безуспешном входе на конкретное устройство. При этом дальнейшие подключения в цикле продолжили бы выполняться.

Проверка версии обеспечения

В современной сетевой инфраструктуре размещено большое количество различных коммутирующих и маршрутизирующих устройств, все они могут быть от различных вендоров со своими сетевыми ОС. Поэтому перед автоматической отправкой конфигурации, необходимо каким-то образом проходить валидацию, иначе команды попросту не будут приняты устройством, которому они не предназначались.

Соответственно формируется задача: даны 3 различных конфигурационных файла, которые нужно отправить соответствующему оборудованию, с выводом на экран информации об отправке.

```
"Импорт необходимых библиотек"
```

```
from getpass import getpass
from netmiko import ConnectHandler
from netmiko.ssh_exception import NetMikoTimeoutException
from paramiko.ssh_exception import SSHException
from netmiko.ssh_exception import AuthenticationException

"Ввод данных для подключения"
username = input("Enter your SSH username: ")
password = getpass()

"Считывание конфигураций в различные переменные"
with open("commands_file_switch") as f:
    commands_list_switch = f.read().splitlines()
with open("commands_file_router") as f:
    commands_list_router = f.read().splitlines()
with open("commands_file_phyrouter") as f:
    commands_list_phyrouter = f.read().splitlines()

"Считывание информации о сетевых устройствах"
with open("devices_file") as f:
    devices_list = f.read().splitlines()

"Запуск цикла с последовательным подключением к устройству"
for devices in devices_list:
    print ("Connecting to device" " + devices) #Вывод на экран
    ip_address_of_device = devices

    "Формирования словаря для подключения"
        ios_device = {
        "device_type": "cisco_ios",
        "ip": ip_address_of_device,
        "username": username,
        "password": password
    }

        "Обработка исключений"
        try:
        net_connect = ConnectHandler(**ios_device)
    except (AuthenticationException):
        print ("Authentication failure: " + ip_address_of_device)
        continue
    except (NetMikoTimeoutException):
        print ("Timeout to device: " + ip_address_of_device)
        continue
    except (EOFError):
        print ("End of file while attempting device " + ip_address_of_device)
        continue
    except (SSHException):
        print ("SSH Issue. Are you sure SSH is enabled? " + ip_address_of_device)
        continue
    except Exception as unknown_error:
        print ("Some other error: " + str(unknown_error))
        continue

    # Формирование списка сетевых ОС"
    list_versions = ["vios_l2-ADVENTERPRISEK9-M",
                     "VIOS-ADVENTERPRISEK9-M",
                     "C1900-UNIVERSALK9-M",
                     "C3750-ADVIPSERVICESK9-M"
                     ]

    # Цикл по проверке сетевой ОС
    for software_ver in list_versions:
        print ("Checking for " + software_ver)
        output_version = net_connect.send_command("show version")
        int_version = 0 # Сброс переменной для цикла
        int_version = output_version.find(software_ver) # Проверка ОС на сетевом устройстве

                # Условие о выводе информации по текущей ОС на устройстве
        if int_version > 0:
```

```
        print ("Software version found: " + software_ver)
        break
    else:
        print ("Did not find " + software_ver)

"Если устройство одной из версий, то отправить соответствующий конфигурационный файл"
    if software_ver == "vios_l2-ADVENTERPRISEK9-M":
    print ("Running " + software_ver + " commands")
    output = net_connect.send_config_set(commands_list_switch)
elif software_ver == "VIOS-ADVENTERPRISEK9-M":
    print ("Running " + software_ver + " commands")
    output = net_connect.send_config_set(commands_list_router)
elif software_ver == "C1900-UNIVERSALK9-M":
    print ("Running " + software_ver + " commands")
    output =      net_connect.send_config_set(commands_list_ph)
elif software_ver == "C3750-ADVIPSERVICESK9-M":
    print ("Running " + software_ver + " commands")
    output = net_connect.send_config_set(commands_list_switch)
print (output)
```

Результат отработки представлен ниже:

```
root@NetworkAutomation-1:~/netmiko+# python3 netmiko6.py
Enter your SSH username: david
Password:
Connecting to device" 192.168.122.72
Checking for vios_l2-ADVENTERPRISEK9-M
Software version found: vios_l2-ADVENTERPRISEK9-M
Running vios_l2-ADVENTERPRISEK9-M commands
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
S1(config)#ip name-server 8.8.8.8
S1(config)#ip domain-lookup
S1(config)#vlan 10
S1(config-vlan)#name Python_VLAN_10
S1(config-vlan)#vlan 20
S1(config-vlan)#name Python_VLAN_20
S1(config-vlan)#vlan 30
S1(config-vlan)#name Python_VLAN_30
S1(config-vlan)#username test password test
S1(config)#router ospf 1
S1(config-router)#network 0.0.0.0 255.255.255.255 area 0
S1(config-router)#end
S1#
Connecting to device" 192.168.122.82
```

Скрипт подключился к устройству, взяв данные для SSH-сессии из файла, определил используемую ОС и отправил конфигурационный файл согласно заданному условию. Также в нем использовались уже изученные обработки исключений, которые позволили не прерываться процессу выполнения программы в случае ошибки, а продолжить свою работу.

Бонусный кейс с подключением без пароля

Подключение к сетевому оборудованию посредством аутентификации с помощью логина/пароля не всегда удобно и безопасно, даже если мы используем модуль getpass(). Netmiko дружит с SSH-ключами и на простом примере, мы покажем как сделать такое подключение.

Но для начала очень кратко по теории :

SSH-ключи представляют собой пару - закрытый и открытый ключ. Закрытый должен храниться в закрытом доступе у клиента, открытый отправляется на сервер и размещается в файле authorized_keys.

В нашем случае при запуске скрипта, он должен будет получить открытый ключ и авторизоваться за счет сравнения с закрытым ключом на стороне сетевого оборудования.

Процесс генерации ключей уже много лет всем известен, если тема для вас новая, то предлагаю ознакомиться по ссылке.

Мы же перейдем непосредственно к практическому примеру с комментариями:

```
# Импорт библиотеки Netmiko
from netmiko import ConnectHandler

# Объявление переменной с путем до открытого SSH-ключа сервера
key_file = "~/.ssh/test_rsa"

# Объявление словаря с указанием типа аутентификации через ключ
cisco1 = {
    "device_type": "cisco_ios",
    "host": "cisco1.lasthop.io",
    "username": "testuser",
    "use_keys": True,
    "key_file": key_file,
}

# Создание подключения
with ConnectHandler(**cisco1) as net_connect:
    output = net_connect.send_command("show ip arp")

# Вывод на экран результата
print(f"\n{output}\n")
```

Вместо заключения

Сегодня мы подробно рассмотрели практические примеры использования Python с Netmiko. Надеюсь, что после двух статей об этой замечательной библиотеке у вас появились поводы, чтобы заняться автоматизацией и перестать ее бояться, если вы еще этого не делали. В конце хотелось бы оставить ссылки на ресурсы по Netmiko, которые вам помогут при изучении:

1. Основная страница Netmiko.
2. Примеры с Netmiko.
3. Практические кейсы для использования.
4. Netmiko и Python2.

Мы же планируем и дальше вас знакомить с новыми инструментами автоматизации, увидимся в новой статье!