**Netmiko 5: Using Passwords with get\_pass()**

It is usually best practice not to store your passwords in Python or JSON. So, what if you want to prompt the user for passwords? How can we have the user enter the password without it being shown on the screen?

**Objectives - Part 1: Using get\_pass()**

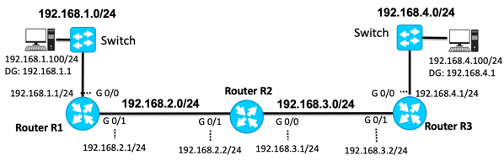
* Prompt the user for a SSH username and password with password not displayed
* Prompt the user for privileged exec password with password not displayed

**Requirements**

* Python: You should be familiar with the basics of Python.
* Using the topology below or your own topology, have access to one or more Cisco IOS devices using SSH.

**Topology**

This lab uses the following topology. You can use a different topology, just be sure to make any necessary modifications to interfaces and addresses. See Netmiko Lab 0: Using a Script for Initial Setup



**Part 1: Using passwords with get\_pass**

**Objectives**

* Prompt the user for a SSH username and password with password not displayed
* Prompt the user for privileged exec password with password not displayed

**Step 1: Continue with the previous JSON file and make the following deletions**

Edit the previous JSON file and name it **my\_devices\_nocred.json.**

**Or**

Create a new text file with the following information, and Save the file as **my\_devices\_nocred.json.** Do not include the information in red.

JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. JSON represents objects as key/value pairs, just like a Python dictionary.

You will notice that our JSON file looks almost identical to our Python dictionary, without the variables. You will also notice that the JSON file is a list of dictionaries, the same dictionaries we used in our previous Python program. So yes, you can copy most of the information from the previous Python program and just make some edits!

Delete all key-value pairs of:

**"username": "admin",**

**"password": "cisco"**

Your **my\_devices\_nocred.json** file should now look like this:

[

{

"ip": "192.168.1.1",

"device\_type": "cisco\_ios"

},

{

"ip": "192.168.2.2",

"device\_type": "cisco\_ios"

},

{

"ip": "192.168.3.2",

"device\_type": "cisco\_ios"

},

{

"ip": "192.168.1.5",

"device\_type": "cisco\_ios"

},

{

"ip": "192.168.4.5",

"device\_type": "cisco\_ios"

}

]

**Step 2: Start with the following code from one of our previous programs**

You should be familiar with this code by now.

import netmiko

print('\n')

for device in devices:

try:

connection = netmiko.ConnectHandler(\*\*device)

print(device['ip'])

print('-'\*11)

output = connection.send\_command('show clock')

print(output)

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device)

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device)

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device)

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

**Step 3: import JSON**

Following **import netmiko**, add the JSON library.

**import json**

**Step 4: import getpass**

Following **from getpass import getpass**, add the getpass from the getpass library.

Note: If we just used **import getpass** then we would have to use **getpass.getpass()** when using this method.

**from getpass import getpass**

**Step 6: Define the function get\_ssh\_credentials()**

You may already be familiar with functions. If not, functions allow you to define a block of code that can be reused multiple times throughout your program. This promotes modular and organized code structure. Instead of duplicating the same code in multiple places, you can define a function once and call it whenever needed.

Following the code in the previous step add the following code:

**def get\_ssh\_credentials():**

**# Username entered is shown in clear text**

**username = input('Enter SSH username: ')**

**# Password entered is not shown**

**password = getpass('Enter SSH password: ')**

**return username, password**

Using a typical **input()** function, the username entered will be displayed in clear text and assigned to the **username** variable. However, **getpass()** is used to enter the password and does not display the password when entered. The password entered is assigned to the variable **password**.

When the get\_ssh\_creditials() function is called, both the username and password values entered by the user will be passed to the variables that used when calling this function.

**Step 6: Define the function get\_secret()**

We will use the same get\_pass() method when the user enters a privileged secret password. This will also be a separate function.

**def get\_secret():**

**# Password entered is not shown**

**secret = getpass('Enter privileged exec password: ')**

**return secret**

The password entered is assigned to the variable **secret**. When the get\_secret() function is called, the value of **secret** will be passed to the variable used when calling this function.

**Step 7: Call the function get\_ssh\_credentials()**

Next, we call the get\_ssh\_credentials() function and assign the returned values to the variables **username** and **password**. For simplicity we have used the same name as the variables in the function.

**username, password = get\_ssh\_credentials()**

**Step 8: Call the function get\_secret()**

First we ask the user if they need a secret password. Not all operations require privileged level access. This can be useful if we modify this program or even better, if we put these two functions in their own program file. Then we can use these functions in any program we write.

**exec\_mode = input('Do you need privileged exec mode? (y/n) ')**

Now, call the get\_secret() function and assign the returned value to the variable **secret**. Once again, for simplicity we have used the same name as the variables in the function.

**if exec\_mode == 'y' or exec\_mode == 'Y':**

**secret = get\_secret()**

**Step 9: Open the JSON file**

Add the following statement:

**with open('my\_devices\_nocred.json') as device\_file:**

**devices = json.load(device\_file)**

Here's a breakdown of what the code does:

* The **open()** function is used to open the "my\_devices\_nocred.json" file in read mode. This function returns a file object that represents the opened file.
* The **with** statement is used to ensure proper handling of the file. It automatically takes care of closing the file after its block of code is executed.
* Inside the with block, the **json.load()** function is used to load the contents of the file into a Python data structure. This function takes the file object returned by **open()** as an argument and parses the JSON data.
* The parsed data is then assigned to the variable **devices**.
* After executing this code, the **devices** variable will hold the contents of the "my\_devices\_nocred.json" file, which is expected to be a list of dictionaries, with each dictionary representing a different device.

**Step 10: Adding entered values to the dictionary items**

Now let's go inside our **for** loop. As shown below add the following code between the **for** and **try** statements:

This will add the **username**, **password** and optional **secret** password to each dictionary items. This way this information is not store in the file and can change depending on our situation,

for device in devices:

**device['username'] = username**

**device['password'] = password**

**if exec\_mode == 'y' or exec\_mode == 'Y':**

**device['secret'] = secret**

try:

**Step11: Display the program**

import netmiko

import json

from getpass import getpass

def get\_ssh\_credentials():

username = input('Enter SSH username: ')

password = getpass('Enter SSH password: ')

return username, password

def get\_secret():

secret = getpass('Enter privileged exec password: ')

return secret

username, password = get\_ssh\_credentials()

exec\_mode = input('Do you need privileged exec mode? (y/n) ')

if exec\_mode == 'y' or exec\_mode == 'Y':

secret = get\_secret('Enter privileged exec password: ')

with open('{path}my\_devices\_nocred.json') as device\_file:

devices = json.load(device\_file)

print('\n')

for device in devices:

device['username'] = username

device['password'] = password

if exec\_mode == 'y' or exec\_mode == 'Y':

device['secret'] = secret

try:

connection = netmiko.ConnectHandler(\*\*device)

print(device['ip'])

print('-'\*11)

output = connection.send\_command('show clock')

print(output)

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device)

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device)

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device)

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

**Step 12: Run the program**

Enter username: **admin**

Password: <cisco password not shown>

Do you need privileged exec mode? (y/n) **n**

-------------------------------------------------------------------------------

Connecting to device 192.168.1.1

\*15:34:26.643 UTC Fri Jun 23 2023

-------------------------------------------------------------------------------

Connecting to device 192.168.2.2

\*12:55:48.791 UTC Fri Jun 23 2023

-------------------------------------------------------------------------------

Connecting to device 192.168.3.2

\*21:11:38.859 UTC Fri Jun 23 2023

Timeout occurred to 192.168.1.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

Timeout occurred to 192.168.4.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

**Step 13: Modify the access level required for a show command**

The **show clock** command only requires user mode access. Modify the **send\_command** method to display the running-config, which requires privileged level access.

**print(connection.send\_command('show running-config'))**

**Step 14: Run the program again but answer 'y' for privileged mode**

Enter username: admin

Password:

Do you need privileged exec mode? (y/n) y

Password:

-------------------------------------------------------------------------------

Connecting to device 192.168.1.1

Building configuration...

Current configuration : 1989 bytes

!

! Last configuration change at 13:44:30 UTC Fri Jun 23 2023 by admin

version 15.2

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname R1

!

Etc.

-------------------------------------------------------------------------------

Connecting to device 192.168.2.2

Building configuration...

Current configuration : 2165 bytes

!

! Last configuration change at 12:25:01 UTC Fri Jun 23 2023 by admin

!

version 15.4

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname R2

!

Etc.

-------------------------------------------------------------------------------

Connecting to device 192.168.3.2

Building configuration...

Current configuration : 1591 bytes

!

! Last configuration change at 17:27:37 UTC Fri Jun 23 2023

!

version 15.4

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

!

hostname R3

!

Etc.

Timeout occurred to 192.168.1.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

Timeout occurred to 192.168.4.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

**Part 2: Importing our get\_credentials code**

Create a new Python file and save it as **my\_tools.py**

from getpass import getpass

# Prompts and returns username and password (not shown)

def get\_credentials():

# Username entered shown in clear text

username = get\_input('Enter SSH username:')

# Password entered not shown

password = getpass('Enter SSH password: ')

'''

'''

Option for having user enter password twice

password = None

while not password:

password = getpass('Enter SSH password: ')

password\_verify = getpass('Retype your password:')

if password != password\_verify:

print('Passwords do not match. Try again.')

password = None

'''

return username, password

In the main program, in the same section as import netmiko, add:

**import my\_tools**

Change:

**username, password = get\_ssh\_credentials()**

to

**username, password = mytools.get\_credentials()**