Lab 01 - Initial Python and Bash Scripting

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Goal: The goal of this lab is to

* Gain familiarity with python and Bash Scripting
* Understand sub-netting concepts

Deliverables: Lab Writeup, cidr.py, screen shot (Note Lab Writeup must be turned in docx or pdf). Do not zip files

Work in teams of 2 (or 3 max). Please work at the pace of your lab partner. Each of you is responsible for doing every step and handing in your own deliverables. Print your responses (at > ) in **BOLD BLUE** text. Put corrections in **BOLD RED**

For lab 2, you will need both the Kali Linux and the Windows 7 images. You should copy these images for use on your laptops. (Assuming you have 8Gb+ RAM). Go to your UC Denver Imagine account (e5.onthehub.com…)

* The PCs in the lab have been set up with Python 2.7 and 3.6 along with pycharm. On laptops, you can load python 3.6 by going to python.org. Then load pycharm from jetbrains. Many of you have used the Kali Linux a lot, meaning the following will be quick for you. If not, pay attention not only to what you are typing in, but what it is doing.
* For today, we will have you installing and using Pycharm inside the VMware image. However in the future, you may go “old school” and use a text editor and compile (.py file) with python zzz.py or python3 zzz.py
* Go to VMware Workstation
* File->Open (virtualmachines directory) Kali 2018.2
* Click on the Kali 2018.2 and go to Edit Virtual Machine Settings->Network and set to **NAT**
* Increase the memory to 8-12GB if you are on the PCs, about 60% of memory on other laptops
* Power on the Kali 2018.2 virtual machine. Log in as **student** with PW: **Student123**
* Go to view->Autosize->Autofit Guest
* Right click on the desktop and **Open Terminal**
* **su** (password: **toor)**
* **ifconfig**
* **ifdown eth0** verify your IP address. If it is 192.168.10.2
* (that’s a zero)…stands for interface down on the first ethernet connection
* **ifup eth0**
* Verify that you have both python versions installed
* **python**
* >>>**quit( )**
* **python3**
* >>>**quit( )**
* On the left hand side, click on the Firefox icon
* Go to <https://www.jetbrains.com/pycharm/download/#section=linux>
* Download the Community Edition
* **cd /home/student/Downloads**
* **tar -xzvf pycharm** (then hit the tab key to complete)
* move the pycharm-community…folder
* **ls**
* **mv pycharm-community… /opt/pycharm**
* make simlinks
* **ln -s /opt/pycharm/bin/pycharm.sh /usr/local/bin/pycharm**
* **ln -s /opt/pycharm/bin/inspect.sh /usr/local/bin/inspect**
* **exit**
* **whoami** (ensure you are student)
* **cd**
* **pycharm**
* Set up your color scheme
* Create a new project. Name it yourlastnameLab01 (then expand project interpreter)
* Choose Python2.7 as the interpreter. We will usually use Python 3.6 but it is important to know that there are syntax differences, so if you are using code examples, it is key to know which version of the interpreter is to be used.
* Right Click and Create a new Python File. Call it yourlastnameLab01
* Print your name, with **print "Your Name"** , then run it
* Now go to **Run->Edit Configurations**. Change your Python interpreter to **Python 3.5.** Once again try **print "Your Name"** then run it. You will see a syntax error. There are a few syntax changes between 2.7 and 3.5. As we go through this course you will have to translate many of these functions and libraries. Most 3rd party libraries will require a manual conversion, but there is a conversion script that will do the basics for you. At the far lower left you will see a square. Click on that and go to Terminal. At the terminal prompt run **2to3 -w .** (little w says write and the . says all python files in the directory)
* 2to3 is said to convert a lot of the built in functions, but doesn't always work with 3rd party libraries.
* Now do an ls and find that it overwrote the file but made a .bak file if it totally messed things up (make sure to use a little w)
* Close out of the virtual machine, by shutting it down. Do not just close out, or you may corrupt your linux machine.

**Part II - Network CIDR Notation - Python**

* In your own words describe CIDR Notation
* In your own words describe the pseudo-code for creating a subnet, given a CIDR notation
* Type in the following code. Do NOT cut and paste. You will learn far more by typing information
* Execute this with 192.168.10.2/26 on python 3.6. From now on you do not have to use Pycharm, but,
* Add comments to help you understand Python, parsing and the
* Provide a screen shot of the results. On Windows you can use snipping tool. On a Mac, you can Command-Shift-4, hold down mouse and draw area…goes to your desktop) and the python code to the Canvas dropbox for Lab01.

import ipaddress

def parseIP(cidrNotation):

addressSplit = cidrNotation.split('/')

ip = addressSplit[0]

cidrSubnet = addressSplit[1]

ipSplit = ip.split('.')

return (ipSplit, cidrSubnet)

def convertCidrToDec(cidrSubnet):

netmask = ""

for x in range(0, int(cidrSubnet)):

netmask += str(1)

for x in range(0, 32 - int(cidrSubnet)):

netmask += str(0)

netmask1 = netmask2 = netmask3 = netmask4 = ""

for x in range(0, 8):

netmask1 += str(netmask[x])

for x in range(8, 16):

netmask2 += str(netmask[x])

for x in range(16, 24):

netmask3 += str(netmask[x])

for x in range(24, 32):

netmask4 += str(netmask[x])

netmask1 = str(int(netmask1, base=2))

netmask2 = str(int(netmask2, base=2))

netmask3 = str(int(netmask3, base=2))

netmask4 = str(int(netmask4, base=2))

return(netmask1, netmask2, netmask3, netmask4)

def printIPsInSubnet(ip, netmask):

if int(netmask[2]) < 255:

count3 = 255-int(netmask[2])

for loop3 in range (0, count3):

for loop4 in range (1,256):

print("{}.{}.{}.{}".format(ip[0], ip[1], loop3, loop4))

elif int(netmask[3]) < 255:

count4 = 255-int(netmask[3])

for loop4 in range (1, count4):

print("{}.{}.{}.{}".format(ip[0], ip[1], ip[2], loop4))

def main():

print("Your Name, Python - CIDR Notation")

cidrNotation = input("Enter your network in CIDR. e.g 192.168.10.0/30: ")

parseString = parseIP(cidrNotation)

ip = parseString[0]

cidrSubnet = parseString[1]

print("Using my own functions")

print("IP:{}.{}.{}.{}\tCIDR:{}".format(ip[0], ip[1],ip[2],ip[3],cidrSubnet))

netmask = convertCidrToDec(cidrSubnet)

print("subnet: {}.{}.{}.{}".format(netmask[0],netmask[1],netmask[2],netmask[3]))

print("IPs in subnet:")

printIPsInSubnet(ip, netmask)

# Let the ipaddress library do the work for you

print("\nUsing import ipaddress:")

network = ipaddress.ip\_network(cidrNotation)

print("IP:{}\tCIDR:{}".format(network, cidrSubnet))

print("subnet:{}".format(network.netmask))

print("IPs in subnet:")

for hosts in network.hosts():

print(hosts)

if \_\_name\_\_ == "\_\_main\_\_":main()

* Turn in the FULLY Commented CIDR.py and a screen shot of it in operation.
* In your own words, summarize 10 lessons that you can use in further cybersecurity programming (Syntax, basics of python, import library, troubleshooting, subnetting stuff, things you had to look up to help you document the code).