Daniel Roh

CMSC 491/791 Active Cyber Defense HW 1

Instructions

- This assignment is for your benefit and should give you a good idea of whether or not this is the class for you.
    - If you struggle with any portion of this assignment, don't hesitate to reach out for help, but understand this is **prerequisite** knowledge that you should have before coming into this class.
- Feel free to use Google/StackOverflow/whatever online resources you deem necessary, just know these are not a replacement for actually understanding the required material.
- After this week, it is expected that you know everything covered in this lab. We will not be answering questions on any of this material after the add/drop date.

Due: September 11, 2019 at 7:00pm

Part 1: Virtual Machine Setup

A virtual machine (VM) is an emulation of a computer system and provide the functionality of a physical computer. They allow us to test configurations and exploits with less risk of harming physical devices. Setting up a VM requires a computer with at least 4 GB of RAM.

Installation:

- Download Virtualbox from: https://www.virtualbox.org/wiki/Downloads
- Download the Ubuntu 18.04.3 LTS .iso file from https://ubuntu.com/download/desktop
- Follow the instructions for setting up an Ubuntu VM in Virtualbox: https://itsfoss.com/install-linux-in-virtualbox/
- Make sure to take a snapshot of the VM after installing it!

Part 2: Introduction to Linux (50 points)

Unlike Windows, Linux is mostly accessed using the terminal with commands. It's important to get familiar with the basics of the operating system before delving deeper. This exercise will familiarize you with some basic Linux commands as well as users, groups, and permissions on a Linux system.

1) Create a group on your Ubuntu VM named "umbc". What command did you use to do this? (4 pts)

“sudo adduser umbc”

2) What is the full path of the file used to store information about each group on a Linux system? (5 pts)

“/etc/group”

3) Open the file from question 2. What is the GID of the umbc group? (4 pts)

1001

4) Use the adduser command to create the “truegrit” user on your Ubuntu VM. Provide a password and leave the rest of the details blank. What is the full path of the file used to store information about each user account on a Linux system? (5 pts)

“/etc/passwd”

5) Open the file from question 4 and use it to answer the following questions: (5 pts)

a. **What is the truegrit user’s UID?** 1002
b. **What is the truegrit user’s GID?** 1002
c. **What is the truegrit user’s default shell?** “/bin/bash”
d. **What is the truegrit user’s home directory?** “/home/truegrit”

6) Add the truegrit user to the umbc group. What command did you use to do this? (4 pts)

“sudo usermod -a -G truegrit umbc”

7) Switch to the truegrit user. Create a file named “test.txt” in truegrit’s home directory. Change the group ownership of test.txt so that it belongs to the umbc group. What command did you use to change test.txt’s group ownership? (4 pts)

“su” then “chown truegrit:umbc test.txt”

8) Change the permissions of test.txt so that the truegrit user has read and write permissions, users in the umbccd group have read permission, and any other users have neither read, write, nor execute permission. What command did you use to do this? (5 pts)

“Chmod 640 test.txt”

9) What is the GID of the sudo group? Which users are members of the sudo group? (4 pts)

27, Only the main account which I called “cyber” is part of the group which I assume is the admin only

10) Edit the umbc group so that it has the same GID as the sudo group. How did you do this? (5 pts)

1st: “groupmod -g 9999 sudo”

2nd: “groupmod -g 27 umbc”

11) After changing the umbc group’s GID, attempt to execute a sudo command as the truegrit user. Was it successful? Why or why not? (5 pts)

No, while the GID for sudo is 27, it seems that it the GID follows the command. So in this case sudo’s GID is now 9999 which is required for the sudo command to work.

Part 3: Basic Programming (50 Points)

Programming is a necessary skill for many aspects of cybersecurity. This exercise will help you practice basic Python programming. As a part of the exercise, you will use the Python socket library to communicate with a program running on a remote server. This program prompts the user to solve a series of addition problems. When all of the problems are solved, the program outputs a flag. If input is not provided quickly enough, the program exits. You can connect to it using the following Linux command:

```
nc umbccd.net 8000
```

It would be nearly impossible for a human to solve these questions quickly enough by hand. Instead of connecting to the remote server over netcat, write a Python script that connects to it using the socket library (Either Python 2 or 3 is fine). The script should receive the addition problems over the socket, perform computations, and send the results back over the network. You will receive a flag if a sufficient number of computations are sent in time.

Documentation for the Python socket library can be found at https://docs.python.org/3/library/socket.html.

1) What is the flag? (15 pts)

DawgSec{numb3rz_@r3_c00l}

2) In a few sentences, describe the design choices you made while writing this script. What was the most challenging part of this exercise? (10 pts)

While writing the script, I was not sure of the amount of data that the server will give me (ie, If the line or lines will be questions, text, or other). So I started by making a script that can pull the question from a set of data given. As I started to solve the problem I noticed that the data format changed after a set of questions, so I needed to adjust my parser to accommodate for multiple types of data formats being sent. I also looked into figuring out how to tell if the flag was given and initially thought it was a fixed number of questions that was given. After a few goes, I realised that it was a random number of questions and that the flag is given in a 3 line set of data.

The most challenging part of the exercise was figuring out the cause for the server time out's. Turns out that the answer format required a newline in the answer but I thought it was due to inefficient code.

3) Provide the source code for your Python script below. (25 pts)

```
#HW 1

#CSMC 491 Active Cyber Defence

#Daniel Roh (FY03817)


import socket
import sys

import time




#---------------------------------------

#Get data from server function

#---------------------------------------

def getData():

        data = s.recv(1024)

        #data = s.recv(1024, socket.MSG_WAITALL)

        #print(data)

   # while data == bytes(b' '):

   #     data = s.recv(1024)

   #     print(data)


        return data


#---------------------------------------

#Parser Function

#---------------------------------------

def parse(data):

        data = data.decode("utf-8") #Decode

        temp = data.splitlines() #Split by lines for processing

        x = 0

        y = 0

        a = 0

        b = 0
```

```
if len(temp) == 3: #If the flag is found

return -2


while not (temp[x][0].isnumeric()): #Find where the first number starts

print(temp[x]) #Print out output in order

x = x + 1

if x >= len(temp):

print("Odd data sent, closing connection(1)")

x = -1

break


if x == -1: #If somthing went wrong

s.close()

sys.exit()


print(temp[x]) #Print out last data


if len(temp[x]) < 8 and len(temp) < 2: #If a answer

return -1

elif len(temp) >= 2 and len(temp[0]) < 8: #If a answer and a question if combined

find = temp[1]

else: #If data is at x

find = temp[x]


while not (find[y].isspace()): #Find where the first number ends

y = y + 1
```

```
        if y > len(find):
        print("Odd data sent 2, closing connection(2)")
        y = -1
        break
        if  y == -1: #If somthing went wrong
        s.close()
        sys.exit()

        a = y + 3 #Starting point of the second value
        b = a;

        if a > len(find): #If only a answer is found
        print("ONLY ANSWER")
        return -1

        while not (find[b].isspace()):
        b = b + 1
        if b > len(find):
        print("Odd data sent 3, closing connection(3)")
        b = -1
        break
        if b == -1: #If somthing went wrong
        s.close()
        sys.exit()

    #DEBUG
    # print(find[x])
```

```
   # print("values" + str(x) + " " + str(y) + " " + str(a) + " " + str(b))

   # print(find[x][0:y])

   # print(find[x][a:b])


        return solver(int(find[0:y]), int(find[a:b]));


#--------------------------------------

#Question solver 90000 function

#--------------------------------------

def solver(number, number2):

        return number + number2




#========================================

#Main Starts here

#========================================

try:

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        print ("Socket created")

except socket.error as err:

        print("Socket failed %s" %(err))


port = 8000


try: #attempt to connect to the server

        host_ip = socket.gethostbyname('umbccd.net')

except socket.gaierror:
        print ("unable to get host")

        sys.exit()


s.connect((host_ip, port))

print ("connected to %s" %(host_ip))

time.sleep(0.1) #Quick pause to let server catch up


going = 1

while not going == 0:

        data = getData()

        sol = parse(data)


        if not sol == -1 and not sol == -2: #If a question is found

        solu = (str(sol) + "\n" ).encode("utf-8")

        s.send(solu)

        print("Answer = " + str(sol)) #debug

        if sol == -2: #If flag is found

        data = data.decode("utf-8")

        temp = data.splitlines()

        print("\n\n ------------------ \n\n")

        print(temp[1])

        print(temp[2])

        print("\n\n ------------------ \n\n")

        going = 0 #Stop looping


s.close()

print("connection closed")
```