Jack Berberian

CIS 245

Networking Assignment

5/2/2022

Contents

[Script can be found here: 1](#_Toc102413568)

[Some things to know before running the networkingscript.sh file: 1](#_Toc102413569)

[Prerequisites: 2](#_Toc102413570)

[Commands that I’ve Chosen to Put into the Script and Why: 3](#_Toc102413571)

[Citation for Script: 4](#_Toc102413572)

[Sample Script Output for Ubuntu and CentOS 5](#_Toc102413573)

[1. CentOS 6](#_Toc102413574)

[2. Ubuntu 6](#_Toc102413575)

[Script: 7](#_Toc102413576)

# Script can be found here:

<https://github.com/jackfrosttt/CIS245-Linux-Administration-Classwork/tree/main/Networking>

# Some things to know before running the networkingscript.sh file:

1. No special restrictions on folder location. You can have the script file in any directory that you want.
2. The script saves the output to the location /tmp/logs/networkingscriptlogoutput/networkingscriptoutput.txt. If the file and or directory doesn’t already exist, it’ll create them by itself.
3. In the directory that the script file is in run this command to make the script executable:
   1. chmod +X networkingscriptoutput.txt
4. Then to run the script file, enter this command:
   1. ./networkingscriptlogoutput.txt
5. The display will output to the command line and also output to /tmp/logs/networkingscriptlogoutput/networkingscriptoutput.txt file and append the file so it’ll save every instance you run the script in case the results need to be compared against.

# Prerequisites:

1. You’ll **need** root or sudo privileges on the system if you don’t have it already. If you are the main system administrator account, in the directory that the file is in, enter the command below, if not, please contact your system administrator. (sudo command) lets you temporarily have root privileges to enter/use any commands you’d like.
   1. To run file with sudo privileges, enter: sudo ./networkingscript.sh
   2. It’ll then prompt you for your password.
2. The script will create the directory and file if it doesn’t exist already at
   1. /tmp/logs/networkingscriptlogoutput/networkingscriptoutput.txt
3. It’s advised to update your system before running the script by entering either (depending on the system):
   1. Ubuntu: sudo apt update && sudo apt upgrade -y
   2. CentOS: sudo apt update && sudo apt upgrade -y
4. You’ll need to install traceroute on the system if you don’t have it already.
   1. On Ubuntu enter: sudo apt install traceroute -y
   2. On CentOS enter: sudo dnf install traceroute -y
5. If you don’t have ping already enter:
   1. On Ubuntu: sudo apt install iputils-ping -y
6. If you don’t have netstat already enter:
   1. On Ubuntu: sudo apt install net-tools -y

# Commands that I’ve Chosen to Put into the Script and Why:

While there are a lot of valuable commands that I could’ve chosen to add into the script file, the main ones that I picked were ping, traceroute, grep “nameserver” /etc/resolv.conf, netstat -tulpn, and ip address | grep …., mostly due to the value they provide for their utility.

One of the main reasons that I picked ping was for its versatile network troubleshooting usage. It can serve dual purposes for troubleshooting external sites as well as internal network systems. A great feature of ping is that it can tell you whether a system is up or offline. If a computer starts to develop network issues, ping can tell whether an issue is inside the network, whether it be a home office or corporate environment. If a website such as Google.com wasn’t loading, one way to diagnose the problem would be to enter: ping google.com. What’d tell me is whether the issue I’m having not reaching Google is with my browser or if it is a network issue. Following that step, I could ping the gateway/router of the network (sometimes 192.168.1.1) and depending on what it returns, I could tell whether there might be an issue with the router. By adding the -c # flag after the ping command and before the target computer, you can limit the ping to the desired number of times as opposed to infinity.

Following up after ping, the reason why I picked traceroute was the utility it provides in connection with the ping command. If have a network issue, depending on what the ping command returns, traceroute will help me determine whether an issue could be internal or external. In a corporate environment, if there are multiple hops in a network, traceroute will help tell where the connection got interrupted. By entering the gateway/router IP address, sometimes 192.168.1.1 (traceroute 192.168.1.1), the traceroute command will trace the connection to the router. If the router is not set up to reject the command and it successfully completes reaches 192.168.1.1, then you’ll be able to confirm that the internal network is okay and the issue must be external. Although if you enter a command such as (traceroute google.com) and it doesn’t complete, you’ll be able to see where it stopped. Overall, traceroute is helpful in identifying whether a network issue is external as opposed to internal as well as identifying where the issue occurs, it could be in a separate internet service providers network.

The main reason that I included grep “nameserver” /etc/resolv.conf was due to the need in identifying the domain name server. The DNS (Domain Name Server) or name server is the network address that the computer asks when visiting websites to translate a website such as Google.com into an IP address that the computer and underlying network will understand. An example of an IP address that could be used to reach Google is 142.250.80.78 and Google’s own DNS server is 8.8.8.8 or 8.8.4.4 for IPv4. In home offices, the DNS is usually automatically provided by the Internet Service Provider or in rare cases, tweaked by the user to a custom one such as Google’s or Cloudflare (1.1.1.1). In corporate environments, one way to restrict website usage would be to setup a custom DNS server on a computer local to network and then have that computer do the lookups instead of each individual computer. On the computer that’s doing the lookups, it’s possible to add blacklists or restrict usage network wide and by entering grep “nameserver” /etc/resolv.conf, it’s easy to tell if the system adopted the network DNS server and if not, which one it is using.

The reasoning behind the command (netstat -tulpn | grep LISTEN) wasn’t more so because of it’s networking diagnostic usage, but because of the security diagnostic ability it can provide. In a corporate environment, if a user ran some unauthorized program and created some sort of server, netstat would be able to identify the program and process number of the port that the user made. Another example of the importance the netstat -tulpn command provides to system administrator is the cyber-security side. Every unnecessary open port creates an additional attack vector that hackers can use to infiltrate the system and cause problems. By closing ports, such as Remote Desktop, file shares, or other programs, it can reduce the options hackers can use to compromise networks. An additional benefit of seeing what ports needed to be closed, is that if a hacker does get into a system, they would be diminished in their ability to exfiltrate the data they do get. In summary, netstat -tulpn is useful for its ability to identify the program that’s listening on either a TCP or UDP port and then it’s port number.

Lastly, one of the most important network commands that I’ve chosen was the ip address command. In the script, ip address | grep -E '((1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])', searches and uses a regular expression to display the IP addresses of the system. The reasoning why this could be considered most valuable is due to just the IP address. On a network, the IP address is one of the most important addressing methods. If a computer also functions as a software defined router, knowing the IP address would be necessary in order to use the aforementioned commands in diagnosing network issues as well as possibly connecting to network file and printer shares that a computer might be hosting.

Overall, out of the many commands, the included ones in this report had the most effective ones in diagnosing some common network issues. In summary, IP

# Citation for Script:

* <https://unix.stackexchange.com/questions/404822/shell-script-to-create-a-file-if-it-doesnt-exist>
* <https://www.cyberciti.biz/faq/unix-linux-check-if-port-is-in-use-command/>
* <https://www.tecmint.com/find-my-dns-server-ip-address-in-linux/>
* <https://www.thegeekstuff.com/2009/11/how-to-execute-ping-command-only-for-n-number-of-packets/>
* <https://kapeli.com/cheat_sheets/Bash_Test_Operators.docset/Contents/Resources/Documents/index>
* <https://linuxize.com/post/bash-if-else-statement/>
* <https://askubuntu.com/questions/1217513/what-does-the-tulpn-option-mean-for-netstat>
* https://unix.stackexchange.com/questions/278939/how-do-you-put-date-and-time-in-a-file-name

# Sample Script Output for Ubuntu and CentOS

1. CentOS

Text

Description automatically generated

1. Ubuntu

Text

Description automatically generated

# Script:

#!/bin/bash

#The variable filename is created and it'll output to networkingscript\_xxxx\_xx\_xx\_##\_##\_AM/PM.txt

filename="networkingscript\_"$(date '+%Y\_%m\_%d\_%I\_%M\_%p')".txt"

#"networkingscript\_"$(date '+%Y\_%m\_%d\_%I\_%M\_%p')".txt"

#if /tmp/logs/networkingscriptlogoutput/"$filename" exists, then do nothing, if it doesn't then

#create the directory then file

if [[ ! -e /tmp/logs/networkingscriptlogoutput/"$filename" ]]; then

mkdir -p /tmp/logs/networkingscriptlogoutput

touch /tmp/logs/networkingscriptlogoutput/"$filename"

echo "File and directory doesn't exist, creating now."

echo "Created directory /tmp/logs/networkingscriptlogoutput"

echo "Created file "$filename""

echo "File and directory doesn't exist, creating now." >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Created directory /tmp/logs/networkingscriptlogoutput" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Created file /tmp/logs/"$filename"" >> /tmp/logs/networkingscriptlogoutput/"$filename"

fi

currenttime=$(date "+%D %T") >> /tmp/logs/networkingscriptlogoutput/"$filename"

currenttime=$(date "+%D %T")

echo "Current time: $currenttime" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Current time: $currenttime"

echo "Filename data stored in = /tmp/logs/networkingscriptlogoutput/"$filename"" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Filename data stored in = /tmp/logs/networkingscriptlogoutput/"$filename""

#Script Runtime Start

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Script Runtime Start - $currenttime" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Script Runtime Start - $currenttime"

#IP Address Start

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "The IP Address(s) for the System are: "

echo "The IP Address(s) for the System are: " >> /tmp/logs/networkingscriptlogoutput/"$filename"

ip address | grep -E '((1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])' >> /tmp/logs/networkingscriptlogoutput/"$filename"

ip address | grep -E '((1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])'

#DNS Start

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "DNS Name Server for the System:"

echo "DNS Name Server for the System:" >> /tmp/logs/networkingscriptlogoutput/"$filename"

grep "nameserver" /etc/resolv.conf >> /tmp/logs/networkingscriptlogoutput/"$filename"

grep "nameserver" /etc/resolv.conf

#Netstat Start

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Output of the TCP and UDP ports that are listening"

echo "Output of the TCP and UDP ports that are listening" >> /tmp/logs/networkingscriptlogoutput/"$filename"

netstat -tulpn | grep LISTEN >> /tmp/logs/networkingscriptlogoutput/"$filename"

netstat -tulpn | grep LISTEN

#Google Ping Start

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Ping to Google.com beginning for 5 times"

echo "Ping to Google.com beginning for 5 times" >> /tmp/logs/networkingscriptlogoutput/"$filename"

ping google.com -c 5 >> /tmp/logs/networkingscriptlogoutput/"$filename"

ping google.com -c 5

#Google Traceroute Start

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Traceroute to Google.com beginning" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Traceroute to Google.com beginning"

traceroute google.com >> /tmp/logs/networkingscriptlogoutput/"$filename"

traceroute google.com

#End of File

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Script finish time: $currenttime"

echo "Script finish time: $currenttime" >> /tmp/logs/networkingscriptlogoutput/"$filename"

#Check File

echo ""

echo "" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Check file /tmp/logs/networkingscriptlogoutput/"$filename" for Script Info" >> /tmp/logs/networkingscriptlogoutput/"$filename"

echo "Check file /tmp/logs/networkingscriptlogoutput/"$filename" for Script Info"