

# **CYB631 Automating Information Security with Python and Shell Scripting**

## **Lab 5: Analyzing logs and network packets using Python**

Your Name: Vaibhav Mayekar

### **Exercises:**

#### **[Environment: Starting with PowerShell ISE]**

1. Windows 10 environment is needed for the lab. Launch Windows PowerShell ISE. Click Start->Run, and then type PowerShell ISE.
2. Now, open a new PowerShell ISE and run as an administrator. Make sure that you navigate to the directory where your script is. Change the executive policy to RemoteSigned.

```
cd C:\Users\cybusr\cyb631  
Set-ExecutionPolicy remotesigned
```

#### **[Exercise I: Export log files]**

3. Let us generate application log for the analysis. First, let us investigate the different types of event logs.

```
Get-WinEvent -ListLog * -ComputerName localhost | Where-Object {  
$_.RecordCount } > winevent.txt
```

```
Get-Content winevent.txt
```

4. Review the contents inside one of the logs from the list. Pick one that has less than 50 entries. For example, you can review the Firewall log and see if you can make sense of the contents in the log.

```
Get-WinEvent -LogName "Microsoft-Windows-AppXDeployment-Server-  
UndockedDeh"
```

5. Briefly describe what Firewall activities that you can observe from the contents in the event log above. \_\_\_\_\_

Paste a snapshot of the results above.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab5 (2)> Get-WinEvent -LogName "Microsoft-Windows-AppXDeployment-Server/Operational"

ProviderName: Microsoft-Windows-AppXDeployment-Server-UndockedDeh

TimeCreated      Id LevelDisplayName Message
-----
10/12/2023 4:32:43 AM 9002 Information Installed EventLog manifest for package: 'M...
10/12/2023 4:32:43 AM 9005 Information Unsetting active EventLog manifest for pack...
10/12/2023 4:32:43 AM 9000 Information Installing EventLog manifest for package: '...
10/12/2023 4:32:43 AM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
10/12/2023 4:32:43 AM 10007 Verbose Encountered package in DEH package evaluati...
9/15/2023 4:47:32 AM 9002 Information Installed EventLog manifest for package: 'M...
9/15/2023 4:47:32 AM 9005 Information Unsetting active EventLog manifest for pack...
9/15/2023 4:47:32 AM 9000 Information Installing EventLog manifest for package: '...
9/15/2023 4:47:32 AM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
9/15/2023 4:47:32 AM 10007 Verbose Encountered package in DEH package evaluati...
9/14/2023 11:54:29 AM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
9/14/2023 11:54:29 AM 10007 Verbose Encountered package in DEH package evaluati...
9/14/2023 11:54:23 AM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
9/14/2023 11:54:23 AM 10007 Verbose Encountered package in DEH package evaluati...
9/6/2023 9:36:37 AM 9002 Information Installed EventLog manifest for package: 'M...
9/6/2023 9:36:36 AM 9005 Information Unsetting active EventLog manifest for pack...
9/6/2023 9:36:36 AM 9000 Information Installing EventLog manifest for package: '...
9/6/2023 9:36:36 AM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
9/6/2023 9:36:36 AM 10007 Verbose Encountered package in DEH package evaluati...
7/28/2023 2:21:36 PM 9003 Information Uninstalling EventLog manifest for package: '...
7/28/2023 2:21:36 PM 9001 Information Uninstalling EventLog manifest for package: '...
7/28/2023 2:21:36 PM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
7/28/2023 2:21:36 PM 10007 Verbose Encountered package in DEH package evaluati...
7/27/2023 10:39:08 AM 9002 Information Installed EventLog manifest for package: 'M...
7/27/2023 10:39:03 AM 9002 Information Installed EventLog manifest for package: 'M...
6/20/2023 6:27:07 PM 9002 Information Installed EventLog manifest for package: 'M...
6/20/2023 6:27:07 PM 9005 Information Unsetting active EventLog manifest for pack...
6/20/2023 6:27:07 PM 9000 Information Installing EventLog manifest for package: '...
6/20/2023 6:27:07 PM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
6/20/2023 6:27:07 PM 10007 Verbose Encountered package in DEH package evaluati...
6/20/2023 10:29:37 AM 9003 Information Uninstalling EventLog manifest for package: '...
6/20/2023 10:29:37 AM 9001 Information Uninstalling EventLog manifest for package: '...
6/20/2023 10:29:37 AM 10002 Verbose Evaluating request for DEH: DEH 'WindowsApp...
```

6. **Get-Eventlog** is similar to **Get-WinEvent** except that the previous one show Windows PowerShell events and the later one records all events.
7. Investigate the results from above. You should see the different Windows event logs as well as the number of entries and the size of each log. We will use logs from **Get-EventLog** as inputs for Python programs later. **Get-EventLog** and **Get-WinEvent** cmdlets can export log entries from the same log but in slightly different data fields. When you analyze them later in Python scripts, please be mindful of the difference and modify the program accordingly if you use logs generated from **Get-WinEvent**.

**Get-EventLog -LogName Application | Export-Csv -Path .\appevent.csv**

8. Investigate the log. Try to understand the meaning of each column in the log.

**cat appevent.csv -head 3**

### [Exercise III: Python3 IDLE and Import Modules]

We will be using Python to analyze the log file since there are many data analytics modules that are useful for the analysis.

9. We will use Python 3 and Python's Integrated Development and Learning Environment (IDLE) for this lab. This instruction and video use Python 3.7.9. The Windows VM on Horizon Desktop has Python 3.10.4. It is fine if you use a different version as long as it is Python 3.
10. If you do not have Python on your own computer, download and install it using the link below. The recent release is Python 3.x. When installing, please make sure that **Add Python 3.x to Path is Check**.

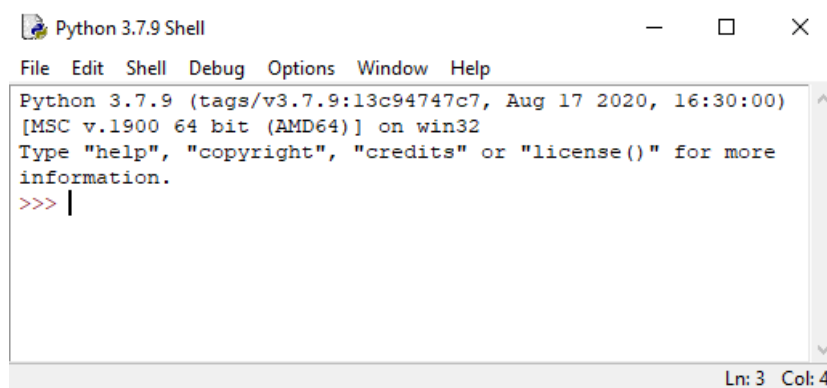
<https://www.python.org/downloads/windows/>



11. Under PowerShell command prompt, use the command below to see the Python version.

```
python --version
```

12. From Windows Start menu, look for Python and click IDLE (Python 3.x) to open it.



13. This is an interactive command tool. It is good for testing simple codes to see if they work. For example, try

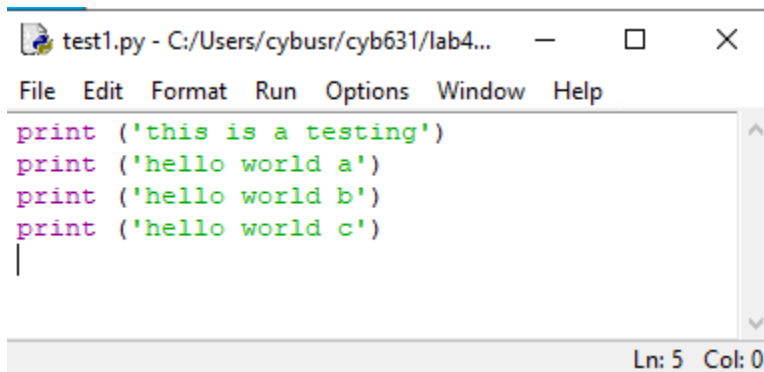
```
print('hello world')
```

```
3+4
```

Paste a screenshot of the results from this step.

```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
print('hello world')
hello world
3+4
7
```

14. Our goal is to create Python script program so that we can run it with other script programs such as PowerShell scripts. So, we will create a file. Click on File, New File. Type the simple Python codes as below and save it as test1.py (File, Save As). Please save it in your work folder, such as C:\Users\lchen\cyb631). Click on Run, Modules. This will show the results in another Python IDLE shell.



```
test1.py - C:/Users/cybusr/cyb631/lab4...
File Edit Format Run Options Window Help
print('this is a testing')
print('hello world a')
print('hello world b')
print('hello world c')
|
Ln: 5 Col: 0
```

15. In addition to the Python IDLE, you can also run the program under PowerShell command prompt. **Important:** Make sure that the directory you run the command has the Python file. If not, navigate to the directory you saved the file before you run the command.

**python test1.py**

16. (Find your file - Only for Horizon Desktop Windows VM) Anything that you save on the desktop is under this path [\\pace.edu\shares\users\lchen\Desktop](https://pace.edu/shares/users/lchen/Desktop) (substitute **lchen** with your login user name). To navigate to this directory:

**cd [\\pace.edu\shares\users\lchen\Desktop](https://pace.edu/shares/users/lchen/Desktop)**

Windows command prompt would display a very long path. You can customize it. For example, using the PowerShell program below to display only the “Desktop” part.

```
function prompt {
    $p = Split-Path -leaf -path (Get-Location)
    "$p> "
}
```

17. There are many existing Python modules that we can utilize. In this lab, we will be using Python modules. To use a module, you will have to install it first before you can import it in the program. We will use three Python modules in this exercise:

Pandas: a data analysis and manipulation tool (<https://pandas.pydata.org/>)

Matplotlib: a data visualization tool (<https://matplotlib.org/>)

Scapy: a packet manipulation tool (<https://scapy.net/>)

18. To import Python module, type the following under PowerShell command prompt. These commands will download and install Python modules from the Internet. Make sure that you run PowerShell as an administrator, or you will not be able to install the Python modules.

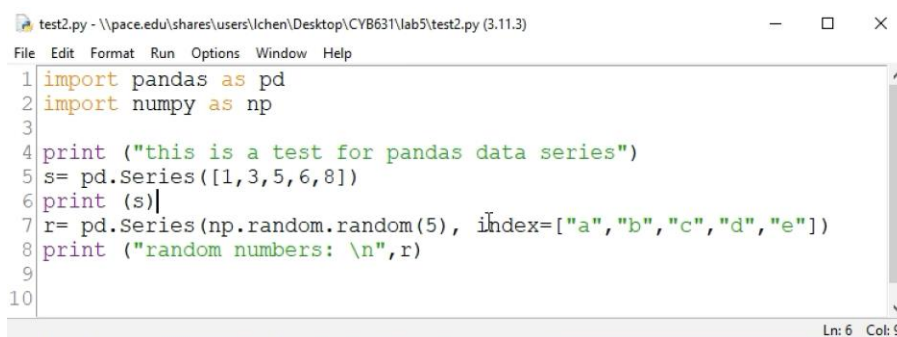
```
python -m pip install pandas
```

```
python -m pip install matplotlib
```

**In the command below, there are two dashes before the command option “pre”.**

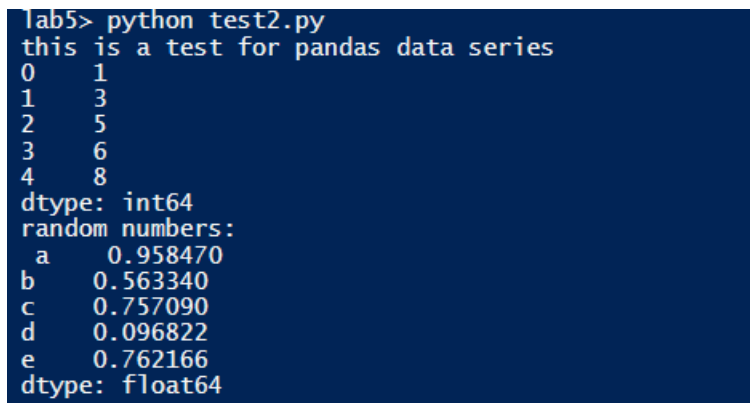
```
python -m pip install --pre scapy[complete]
```

19. To test if Pandas is accessible, modify your testing program to the one below. Run the module. If it is successful, you should not have error messages and the information in the print function will be printed on the output window.



```
test2.py - \\pace.edu\shares\users\lchen\Desktop\CYB631\lab5\test2.py (3.11.3)
File Edit Format Run Options Window Help
1 import pandas as pd
2 import numpy as np
3
4 print ("this is a test for pandas data series")
5 s= pd.Series([1,3,5,6,8])
6 print (s)
7 r= pd.Series(np.random.random(5), index=["a","b","c","d","e"])
8 print ("random numbers: \n",r)
9
10
```

20. Paste a screenshot of your results from the previous step to show that you are able to import pandas module.



```
lab5> python test2.py
this is a test for pandas data series
0    1
1    3
2    5
3    6
4    8
dtype: int64
random numbers:
a    0.958470
b    0.563340
c    0.757090
d    0.096822
e    0.762166
dtype: float64
```

## [Exercise IV: Basic log analysis using Pandas]

21. Using Python IDLE, open **applog.py**. Review the codes and run the file to understand what it does. The program reads the application log (appevent.csv) generated in the previous exercise and group the logs by the **Source** of logs. We are using Pandas module for data processing here.

22. Modify the program to show the logs by **EntryType** and by **Category**.

Show the revised program:

```
import pandas as pd

df = pd.read_csv('appevent.csv', encoding="ISO-8859-1", skiprows=1)

by_source = df.groupby('Source')
print(by_source.size())

by_entry_type = df.groupby('EntryType')
print(by_entry_type.size())

by_category = df.groupby('Category')
print(by_category.size())
```

Run the revised program and paste the results.

```
Tab5> python applogrev.py
Source
.NET Runtime Optimization Service      8
AVLogEvent                             39
Application Error                       70
Application Hang                       12
COM+                                    2
CertEnroll                             36
Chrome                                  2
Desktop Window Manager                 34
Dwminit                                1
ESENT                                 121
EventSystem                           20
ImControllerInstallerService           2
ImControllerService                   27
MSDTC                                  1
MSDTC 2                                2
MSDTC Client 2                         7
Microsoft Office                       3
Microsoft Office 16                   16
Microsoft-Windows-AppModel-State       1
Microsoft-Windows-CAPI2               10
Microsoft-Windows-Defrag              80
Microsoft-Windows-RestartManager      6255
Microsoft-Windows-System-Restore      87
Microsoft-Windows-User Profiles Service 42
Microsoft-Windows-WMI                 148
Microsoft-Windows-Winsrv              1
MsiInstaller                          259
NVIDIA GeForce Experience SelfUpdate Source 1
Office 2016 Licensing Service          1
Outlook                                7
RtkAudioUniversalService              36
SecurityCenter                        718
Software Protection Platform Service  5944
Steam Client Service                  1
System Restore                        147
```

```

System Restore 147
VSS 159
Windows Error Reporting 483
Windows Search Service 75
Windows Search Service Profile Notification 1
WlcIntfy 161
acvpndownloader_major 286
acvpndownloader_minor 21
edgeupdate 260
edgeupdatem 4
gupdate 360
gupdatem 14
dtype: int64
EntryType
0 96
Error 184
Information 12599
Warning 3086
dtype: int64
Category
(0) 15404
(1) 272
(101) 12
(14) 6
(2) 1
(3) 2
Application Crashing Events 70
Browser Events 2
General 59
Logging/Recovery 62
Search service 75
dtype: int64

```

### [Exercise V: Plotting Time Series using Dataframe.plot]

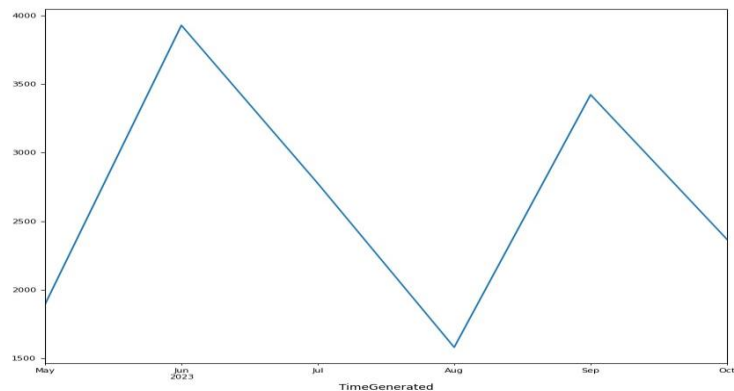
23. Using Python IDLE, open **logfreq.py**. Review the codes and run the file to understand what it does. The program reads the application log (**appevent.csv**) generated in the previous exercise. It extracts TimeGenerated information from the log, converts it to DateTime format in Python, creates a frequency table grouped by Month, and plots the frequency table.
24. Paste a screenshot of the plot.



```

lab5> python logfreq.py
EventID
MachineName
Data
Index
Category
CategoryNumber
EntryType
Message
Source
ReplacementStrings
InstanceId
TimeGenerated
TimeWritten
UserName
Site
Container
      TimeGenerated      TimeWritten
0  10/22/2023  5:57:38 PM  10/22/2023  5:57:38 PM
1  10/22/2023  5:54:24 PM  10/22/2023  5:54:24 PM
2  10/22/2023  5:53:42 PM  10/22/2023  5:53:42 PM
3  10/22/2023  5:52:21 PM  10/22/2023  5:52:21 PM
4  10/22/2023  5:51:53 PM  10/22/2023  5:51:53 PM
      TimeGenerated
0  2023-10-22 17:57:38
1  2023-10-22 17:54:24
2  2023-10-22 17:53:42
3  2023-10-22 17:52:21
4  2023-10-22 17:51:53

```



25. Modify the program to show the number of log entries grouped by day instead of by month.

26. Show the revised program:

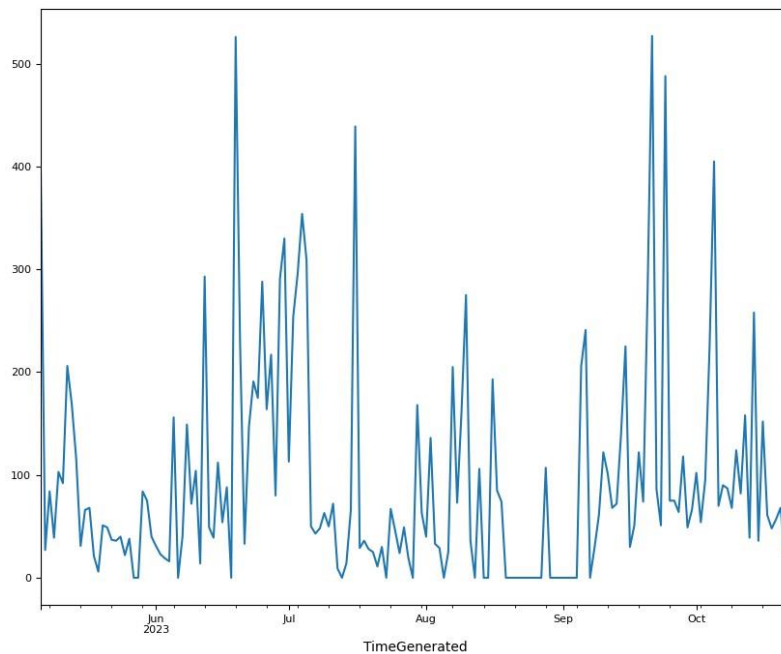
```

import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('appevent.csv', encoding="ISO-8859-1", skiprows=1)
for col in df.columns:
    print(col)
df_date = df[['TimeGenerated', 'TimeWritten']]
print(df_date.head(5))
ts1 = pd.to_datetime(df_date['TimeGenerated'], format='%m/%d/%Y %I:%M:%S %p')
df_ts1 = pd.DataFrame({
    "TimeGenerated": ts1
})
print(df_ts1.head(5))
df_grp = df_ts1.groupby(pd.Grouper(key='TimeGenerated', freq='D')).TimeGenerated.count()
fig, ax = plt.subplots(figsize=(10, 8))
df_grp.plot(kind='line', ax=ax, fontsize=8)
fig.savefig('fig2.png')

```



27. Run the revised program and paste the plot by day.



### [Exercise VI: Sniffing Packets using Scapy]

28. For packet manipulation, Scapy will need Npcap (<https://nmap.org/npcap/#download>) and **please install it**. As an alternative, you can install Wireshark (<https://www.wireshark.org/#download>) which installs Npcap by default. If you do not have Npcap, your program will have error messages like the one below.

```
===== RESTART: C:\Users\lchen\cyb631\lab4\netana.py =====
WARNING: No libpcap provider available ! pcap won't be used
[*] Start sniffing...
Traceback (most recent call last):
  File "C:\Users\lchen\cyb631\lab4\netana.py", line 9, in <module>
    sniff(iface=interface, filter="ip", prn=print_packet)
  File "C:\Users\lchen\AppData\Local\Programs\Python\Python37\lib\site-packages\
scapy\sendrecv.py", line 1263, in sniff
    sniffer._run(*args, **kwargs)
  File "C:\Users\lchen\AppData\Local\Programs\Python\Python37\lib\site-packages\
scapy\sendrecv.py", line 1128, in _run
    **karg)] = iface
  File "C:\Users\lchen\AppData\Local\Programs\Python\Python37\lib\site-packages\
scapy\arch\windows\__init__.py", line 915, in __init__
    "Sniffing and sending packets is not available at layer 2: "
RuntimeError: Sniffing and sending packets is not available at layer 2: winpcap
is not installed. You may use conf.L3socket or conf.L3socket6 to access layer 3
>>>
```

29. Before we start packet manipulation, we will need to specify the network adapter where we are receiving or sending packets. Please use the following cmdlet in Powershell.

### Get-NetAdapter

Paste a screenshot of your results from above:

```
PS C:\Windows\system32> Get-NetAdapter
```

Name	InterfaceDescription	ifIndex	Status	MacAddress	LinkSpeed
Ethernet0	vmxnet3 Ethernet Adapter	10	Up	00-50-56-A3-EE-CA	10 Gbps

30. Using Python IDLE, open **sniff.py**. Review the codes. This program will sniff out 10 IP packets from your network adapter. Replace the network adapter with the one that you would like to sniff packet from.
31. Run the program to understand what it does. For a mapping of protocol number and protocol name, see <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

What is the transport layer protocol used by the IP packets that you sniff out?

### Transmission Control Protocol(tcp)

Paste a screenshot of the results from the sniffing:

```
PS C:\Windows\system32> cd "\\pace.edu\shares\users\vm81403n\Desktop\lab5"
PS Microsoft.PowerShell.Core\FileSystem::\\pace.edu\shares\users\vm81403n\Desktop\lab5> python sniff.py
[*] Start sniffing...
[!]New Packet: 10.1.29.45->192.168.81.63 using 6
[!]New Packet: 192.168.81.63->10.1.29.45 using 6
[!]New Packet: 10.1.29.45->192.168.81.63 using 6
[!]New Packet: 10.1.29.45->192.168.81.63 using 6
[!]New Packet: 192.168.81.63->10.1.29.45 using 6
[!]New Packet: 192.168.81.63->10.1.29.45 using 6
[!]New Packet: 192.168.81.63->10.1.29.45 using 6
[!]New Packet: 10.1.29.45->192.168.81.63 using 6
[!]New Packet: 10.1.29.45->192.168.81.63 using 6
[!]New Packet: 192.168.81.63->10.1.29.45 using 6
[*] Stop sniffing
```

32. Modify the program to sniff only TCP packets and show the TCP port numbers along with the IP addresses such as the format below:

[!]New Packet: 192.168.1.208:50593->172.217.12.174:443 using 6

33. Show the revised program:

```
from scapy.all import *
interface="Ethernet0"
def print_packet(packet):
    ip_layer=packet.getlayer(IP);
    tcp_layer=packet.getlayer(TCP);
    print("[!]New Packet: {src}:{sport}->{dst}:{dport} using {proto}".format(src=ip_layer.src,sport=tcp_layer.sport,dst=ip_layer.dst, dport=tcp_layer.dport, proto=ip_layer.proto));
print("[*] Start sniffing...")
sniff(count=10,iface=interface, filter="tcp", prn=print_packet)
print("[*] Stop sniffing")
```

34. Run the revised program and paste a screenshot of the results.

```

PS Microsoft.PowerShell.Core\FileSystem::\\pace.edu\shares\users\vm81403n\Desktop\lab5> python sniffrev.py
[*] Start sniffing...
[!]New Packet: 192.168.81.63:22443->10.1.29.45:52841 using 6
[!]New Packet: 10.1.29.45:52841->192.168.81.63:22443 using 6
[!]New Packet: 10.1.29.45:52841->192.168.81.63:22443 using 6
[!]New Packet: 10.1.29.45:52841->192.168.81.63:22443 using 6
[!]New Packet: 192.168.81.63:22443->10.1.29.45:52841 using 6
[!]New Packet: 192.168.81.63:22443->10.1.29.45:52841 using 6
[!]New Packet: 10.1.29.45:52841->192.168.81.63:22443 using 6
[!]New Packet: 192.168.81.63:22443->10.1.29.45:52841 using 6
[!]New Packet: 10.1.29.45:52841->192.168.81.63:22443 using 6
[!]New Packet: 10.1.29.45:52841->192.168.81.63:22443 using 6
[*] Stop sniffing

```

## [Exercise VII: Generate Statistics of Network Traffic]

35. Create a Python program to sniff and analyze packets. The programs will have to achieve the following goals:

- Capture at least 5000 network packets while using a browser visiting websites, such as [www.pace.edu](http://www.pace.edu) so that the program can sniff out packets between the browser and the web sites.
- Group and tally source IP addresses in the packets sniffed out. Show the frequencies of unique source IP addresses accessed.
- Group and tally destination IP addresses in the packets sniffed out. Show the frequencies of unique destination IP addresses accessed.
- Timestamp every packet sniffed out, and plot the frequencies of packets over time grouped by either millisecond or second depending on how long it takes for you to capture the packets. If packets are captured within 1 or 2 seconds, showing the frequencies in millisecond or 10 millisecond may make more sense to see the changes over time.

36. To achieve the goals, you can use either one program or multiple programs. Please describe the names of the programs that you develop and the purpose of each program.

```

from scapy.all import sniff, Packet
import time

```

```

def packet_handler(packet):
    if 'www.pace.edu' in str(packet):
        timestamp = time.time()
        with open('captured_packets.txt', 'a') as file:
            file.write(f'{timestamp} {packet}\n')

```

```

sniff(iface='Ethernet0', prn=packet_handler, count=5000)

```

```
print("Packet capture complete.")
```

for analyze packet code:

```
1698354829.4382622 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"
1698354829.5914044 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"
1698355472.9487739 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"
1698362399.529816 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"
1698362399.6741374 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"
1698365858.3341696 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"
1698365858.3808446 Ether / IP / UDP / DNS Qry "b'www.pace.edu.'"

```

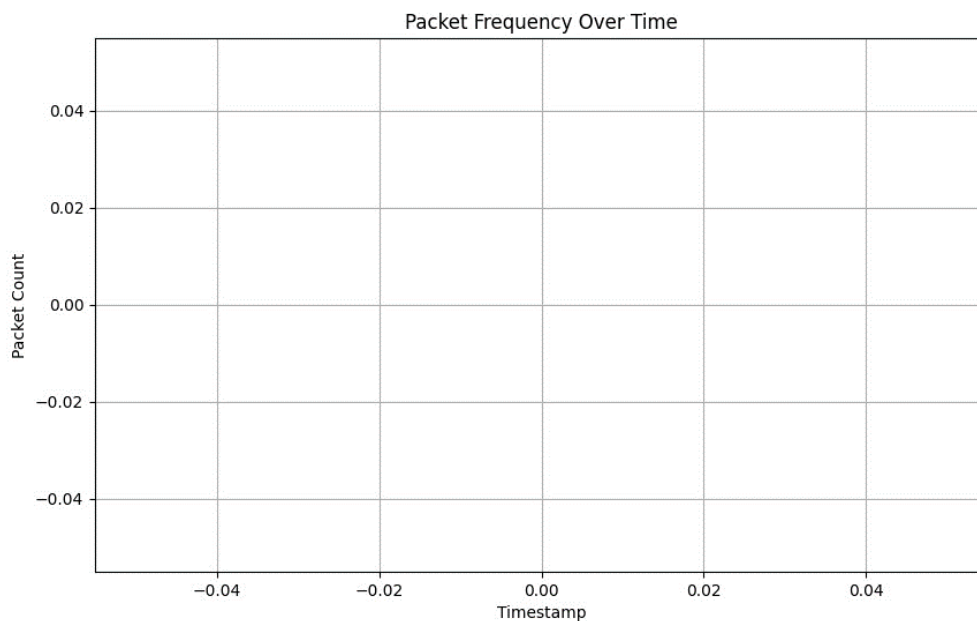
37. Paste program codes here or submit program files as additional submission to the report.

```
from scapy.all import sniff, Packet
import matplotlib.pyplot as plt
import time
def packet_handler(packet):
    if 'www.pace.edu' in str(packet):
        timestamp = time.time()
        with open('captured_packets.txt', 'a') as file:
            file.write(f'{timestamp} {packet}\n')
sniff(iface='Ethernet0', prn=packet_handler, count=5000)
with open('captured_packets.txt', 'r') as file:
    packet_counts = {}
    timestamps = []
    for line in file:
        parts = line.split()
        if len(parts) == 2:
            timestamp, packet_data = parts
            timestamp = float(timestamp)
            timestamps.append(timestamp)
            packet_counts[timestamp] = packet_counts.get(timestamp, 0) + 1

sorted_timestamps = sorted(timestamps)
counts = [packet_counts[timestamp] for timestamp in sorted_timestamps]
```

```
plt.figure(figsize=(10, 6))
plt.plot(sorted_timestamps, counts, marker='o', linestyle='-', color='b')
plt.title('Packet Frequency Over Time')
plt.xlabel('Timestamp')
plt.ylabel('Packet Count')
plt.grid(True)

plt.savefig("fig6.png")
plt.show()
```



38. Show the results of your programs, the frequency tables, and the plot if you choose to do it. If the results are too long, output them in a file and submit the file in additional to your report.

