Title :

NAME OF GROUP MEMBER :

* Avishek: Identified changes in active services/processes
* Vaibhav Mayekar: Identified active services/processes
* Sanketh: Developed a Python program for host log analysis
* Kastubh Kadam: Implemented Windows notifications

**DATE : 12/07/2023**

Basic task 1:

**A diagram of a server and computer

Description automatically generated**

**Basic task 2:**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

Techincal solution :

The script appears to be a PowerShell script used to get service information from a server. It uses the Get-Process cmdlet to retrieve a list of all running processes on the server. Then, it sorts the list by CPU usage in descending order and selects the first 5 processes. For each process, the script uses the Get-WmiObject cmdlet to retrieve the corresponding service object. If the service object exists, the script writes out the following information:

* Process name
* Memory usage
* CPU usage
* Service name
* Service description
* Service status

This script is helpful for troubleshooting performance problems. By identifying the processes that are using the most CPU, you can determine which services are responsible for the high CPU usage.

1] **A script for identifying active services/processes**

A blue screen with white text

Description automatically generated

Report on server

A blue screen with white text

Description automatically generated

Report on windows

2]

A screenshot of a computer error message

Description automatically generated

3]

The Python script provided utilizes the Scapy library to perform network packet sniffing on a chosen network interface. Here's what the script does:

Interface Selection: The script selects a network interface for packet sniffing. There are two lines specifying the interface, but only the second one takes effect. Please choose the appropriate line according to the network interface you wish to use.

Packet Filtering:The script captures only TCP packets. The sniff function is set up with a filter to capture packets with the TCP protocol. You can modify or remove the filter based on your specific needs.

Packet Processing: For each captured packet, the script checks if it has a TCP layer. If it does, the script extracts and prints information about the packet, including source and destination IP addresses, source and destination port numbers, and the protocol (TCP).

Printing Information: The print\_packet function formats and prints the relevant details of each TCP packet.

Execution: The script initiates the packet sniffing process using the sniff function, capturing a specific number of packets (in this case, 10). The captured packets trigger the print\_packet function to display information about each TCP packet.

Permissions and Considerations: Packet sniffing usually requires elevated privileges, so please ensure that the script is executed with the necessary permissions. Additionally, be conscious of legal and ethical considerations when sniffing network traffic, as unauthorized interception may violate privacy and regulations.

**In conclusion, the script is a fundamental packet sniffer that concentrates on TCP traffic. It provides insights into the specifics of the captured packets on the chosen network interface.**

A computer code on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

The purpose of this script is to analyze network traffic by capturing a specified number of packets on a specific interface using Scapy. The captured packets are then processed using a callback function which extracts IP layer information and timestamps each packet. The script maintains two dictionaries to count the occurrences of source and destination IP addresses, and records the timestamps of each packet in a list. Matplotlib is used to create a histogram representing the frequency of packets over time based on their timestamps. The script then prints out the captured source and destination IP address frequencies and displays the histogram. Please note that the network adapter selected in the experiment was Ethernet0, but this can be changed for desired results on a Windows server. Additionally, due to import issues with matplotlib.pyplot, alternative modules such as plotly, seaborn, bokeh or altair can be used. Overall, this script provides valuable insights into network traffic analysis.

A screenshot of a computer program

Description automatically generated A screen shot of a computer code

Description automatically generated  
A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

4]

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated