perplexity

Generate a Python script that extracts all active applications of the local host, their network activities, and track all network traffic, and saves this information to a file with a date and time stamp.

Here's a Python script that extracts details of all active applications, their network activities, and tracks network traffic. It saves this information to a file with a timestamped filename.

Python Script: Active Applications and Network Traffic Monitor

```
import psutil
import datetime
import os
from scapy.all import sniff, IP, TCP, UDP

def get_active_applications():
    """
    Fetches all active processes and their details.
    """
    active_apps = []
    for proc in psutil.process_iter(['pid', 'name', 'connections']):
        try:
            connections = proc.info['connections']
            if connections:
                for conn in connections:
                    if conn.status == psutil.CONN_ESTABLISHED:
                        active_apps.append({
                            "pid": proc.info['pid'],
                            "name": proc.info['name'],
                            "local_address": f"{conn.laddr.ip}:{conn.laddr.port}",
                            "remote_address": f"{conn.raddr.ip}:{conn.raddr.port}" if cor
                            "status": conn.status
                        })
        except (psutil.AccessDenied, psutil.NoSuchProcess):
            continue
    return active_apps

def capture_network_traffic(packet_list, packet):
    """
    Callback function to capture network packets.
    """
    if IP in packet:
        packet_info = {
            "timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
```

```
            "source": packet[IP].src,
            "destination": packet[IP].dst,
            "protocol": packet[IP].proto,
            "size": len(packet)
        }
        packet_list.append(packet_info)

def monitor_system():
    """
    Monitors active applications and network traffic, and saves the data to a file.
    """
    # Generate a unique filename with date and time stamp
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    output_file = f"system_monitor_{timestamp}.txt"

    # Capture active applications
    active_apps = get_active_applications()

    # Capture network traffic for 10 seconds
    print("Capturing network traffic for 10 seconds...")
    captured_packets = []
    sniff(timeout=10, prn=lambda pkt: capture_network_traffic(captured_packets, pkt), sto

    # Save data to file
    with open(output_file, 'w') as file:
        file.write(f"System Monitor Report - {timestamp}\n")
        file.write("=" * 50 + "\n\n")

        # Write active applications
        file.write("Active Applications:\n")
        file.write("-" * 50 + "\n")
        for app in active_apps:
            file.write(f"PID: {app['pid']}, Name: {app['name']}, "
                       f"Local Address: {app['local_address']}, "
                       f"Remote Address: {app['remote_address']}, Status: {app['status']}

        file.write("\n\n")

        # Write network traffic details
        file.write("Captured Network Traffic:\n")
        file.write("-" * 50 + "\n")
        for pkt in captured_packets:
            file.write(f"Timestamp: {pkt['timestamp']}, Source: {pkt['source']}, "
                       f"Destination: {pkt['destination']}, Protocol: {pkt['protocol']},
                       f"Size: {pkt['size']} bytes\n")

    print(f"System monitor report saved to: {os.path.abspath(output_file)}")

if __name__ == "__main__":
    monitor_system()
```

How It Works

1. **Active Applications**:
   - The script uses `psutil` to fetch details of all active processes with established network connections.
   - For each process, it retrieves the PID, name, local address, remote address (if available), and connection status.
2. **Network Traffic Capture**:
   - The `scapy` library is used to sniff packets on the local host for 10 seconds.
   - Captured packets include information such as source IP, destination IP, protocol (TCP/UDP), and packet size.
3. **Save to File**:
   - Both the active application details and captured network traffic are saved to a timestamped text file (e.g., `system_monitor_2025-04-04_22-49-30.txt`).
4. **Output Example**:
   The generated report includes two sections:
   - **Active Applications**:
     ```
     PID: 1234, Name: chrome.exe, Local Address: 192.168.1.100:54321, Remote Address:
     ```
   - **Captured Network Traffic**:
     ```
     Timestamp: 2025-04-04 22:49:30, Source: 192.168.1.100, Destination: 142.250.74.14
     ```

Requirements

1. Install necessary Python libraries before running the script:
   ```
   pip install psutil scapy
   ```
2. Run the script with administrator/root privileges to allow access to network interfaces and process details.

Features

- Monitors active applications with established network connections.
- Captures live network traffic for a specified duration (default is 10 seconds).
- Saves the collected data into a well-organized text file with a timestamped filename.

Customizations

1. **Capture Duration**:
   - Adjust the `timeout` value in `sniff()` to change how long the script captures traffic.
2. **Output Format**:
   - Modify the `file.write()` statements to save data in CSV or JSON format if needed.
3. **Filters**:
   - Use Scapy filters (e.g., `filter="tcp"` in `sniff()`) to capture specific types of traffic.

This script provides a comprehensive snapshot of system activity and network usage for monitoring or troubleshooting purposes!

⁂