perplexity

Exception has occurred: ValueError

invalid attr name 'connections'
File
"C:\Users\jmlad\Desktop\Cybersecurity\Python_Network_Automation\activeApp_and_networkTraffic_localHost.py", line 11, in get_active_applications
for proc in psutil.process_iter(['pid', 'name', 'connections']):
File
"C:\Users\jmlad\Desktop\Cybersecurity\Python_Network_Automation\activeApp_and_networkTraffic_localHost.py", line 51, in monitor_system
active_apps = get_active_applications()
File
"C:\Users\jmlad\Desktop\Cybersecurity\Python_Network_Automation\activeApp_and_networkTraffic_localHost.py", line 84, in <module>
monitor_system()
ValueError: invalid attr name 'connections'

The error `ValueError: invalid attr name 'connections'` occurs because `connections` is not a valid attribute for `psutil.process_iter()` in recent versions of `psutil`. Instead, you should use `net_connections()` for system-wide socket connections or `Process.connections()` for per-process network connections.

Here's the corrected script:

Corrected Script

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

def get_active_applications():
    """
    Fetches all active processes and their network connections.
    """
    active_apps = []
    for proc in psutil.process_iter(['pid', 'name']):
        try:
            # Get network connections for the process
            connections = proc.connections(kind='inet')  # 'inet' filters for internet co
            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 con
                            "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()
```

Key Fixes

1. **Replaced `connections` Attribute**:
   - The `connections` attribute is invalid for `process_iter()`. Instead, the script now uses `proc.connections(kind='inet')`, which retrieves per-process internet connections.
2. **Error Handling**:
   - Added handling for `AccessDenied` and `NoSuchProcess` exceptions to prevent crashes when accessing restricted processes.
3. **Active Applications**:
   - The script now correctly fetches details of processes with established internet connections (`CONN_ESTABLISHED`).

Requirements

1. Install necessary libraries:
   ```
   pip install psutil scapy
   ```
2. Run the script with administrator/root privileges to access process details and sniff network traffic.

Output Example

The generated report will include two sections:

Active Applications:

```
PID: 1234, Name: chrome.exe, Local Address: 192.168.1.100:54321, Remote Address: 142.250.
PID: 5678, Name: python.exe, Local Address: 192.168.1.100:65432, Remote Address: N/A, Sta
```

Captured Network Traffic:

```
Timestamp: 2025-04-04 22:49:30, Source: 192.168.1.100, Destination: 142.250.74.14, Protoc
Timestamp: 2025-04-04 22:49:31, Source: 192.168.1.101, Destination: 8.8.8.8, Protocol: 17
```

Notes

1. **Cross-Platform Compatibility**:
   - Works on both Windows and Linux systems.
   - On Linux systems where root privileges are required for certain operations (e.g., sniffing), ensure you run the script as root.
2. **Traffic Capture Duration**:
   - Adjust the `timeout` parameter in the `sniff()` function to change how long traffic is captured.
3. **Filters**:
   - Add filters to `sniff()` (e.g., `filter="tcp"` or `filter="port 80"`) to capture specific types of packets.

This script provides a comprehensive snapshot of system activity and network usage!

⁂